BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 08161169 A

(43) Date of publication of application: 21.06.96

(51) Int. Cl G06F 9/38
G06F 9/28
G06F 9/45

(21) Application number: 06305907

(22) Date of filing: 09.12.94

(71) Applicant: TOSHIBA CORP

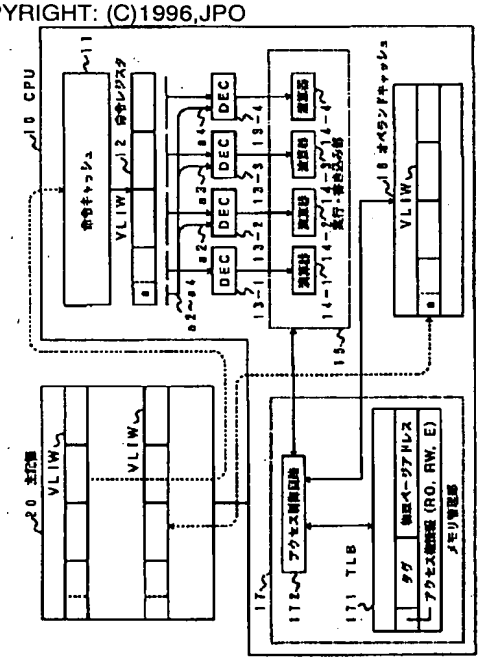
(72) Inventor: TAKEUCHI YOICHIRO

(54) VLIW TYPE COMPUTER SYSTEM AND METHOD FOR INTERPRETING/ EXECUTING VLIW

(57) Abstract:

PURPOSE: To reduce a useless storage area due to a code part conventionally occupied by a nop instruction on an VLIW (very long instruction word).

CONSTITUTION: Istruction validating information (a) consisting of bits a2 to a4 indicating which instruction field (invalid instruction field) does not require interpretation/execution out of a 2nd instruction field and after is formed on a part of the leading field of a VLIW having four instruction fields, and when the VLIW is fetched in an instruction register 12, the contents (excluding the information (a)) of the 1st instruction field are unconditionally interpreted by a decoder 13-1 to control a computing element 14-1 and the contents of the 2nd to 4th instruction fields are interpreted by respective decoders 13-2 to 13-4 in accordance with the states of the bits a2 to a4 in the information (a) to control computing elements 14-2 to 14-4, so that the invalid instruction field part can be utilized as a variable/constant area.



.52#1 ✓

(1)

(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開平８－１６１１６９

(43)公開日　平成8年(1996)6月21日

(51)Int.Cl.⁶		識別記号	庁内整理番号	F I			技術表示箇所
Ｇ０６Ｆ	9/38	３７０　Ｘ					
	9/28	３２０	7230－５Ｂ				
	9/45						
			7737－５Ｂ	Ｇ０６Ｆ	9/44	３２２　Ｅ	
			7737－５Ｂ			３２２　Ｊ	

審査請求　未請求　請求項の数６　ＯＬ　（全２３頁）

(21)出願番号　特願平6-305907

(22)出願日　平成6年(1994)12月9日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　竹内　陽一郎
東京都府中市東芝町1番地　株式会社東芝府中工場内

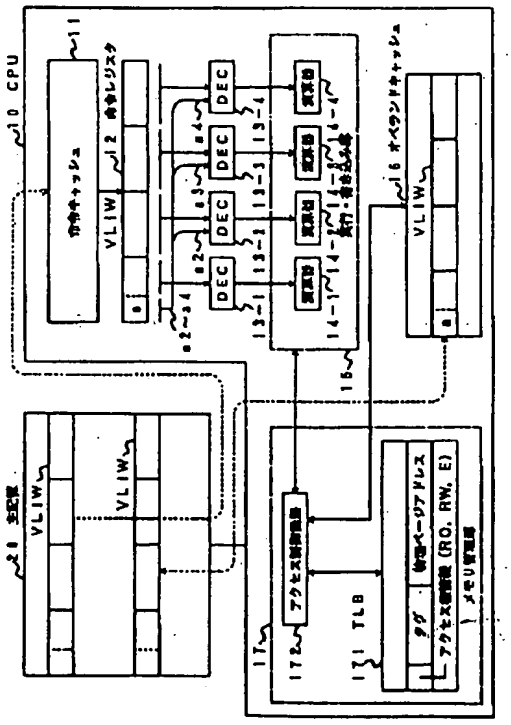
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】ＶＬＩＷ方式の計算機システム及びＶＬＩＷの解釈・実行方法

(57)【要約】

【目的】従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができるようにする。

【構成】４つの命令フィールドを持つＶＬＩＷの先頭フィールドの一部に、２番目以降の命令フィールドのうち、解釈・実行の必要のない命令フィールド（無効命令フィールド）がどれであるかを示すビットａ２～ａ４からなる命令有効化情報ａを設け、当該ＶＬＩＷが命令レジスタ１２にフェッチされた際には、１番目の命令フィールドの（命令有効化情報ａを除く）内容を無条件でデコーダ１３-1により解釈して演算器１４-1を制御すると共に、２番目～４番目の命令フィールドの内容を命令有効化情報ａ中のビットａ２～ａ４の状態に応じてデコーダ１３-2～１３-4により解釈して演算器１４-2～１４-4を制御することで、無効命令フィールド部分を変数／定数領域として利用可能な構成とする。



1

【特許請求の範囲】

【請求項１】　並列動作可能な複数の演算器と、

これら各演算器を個々に制御するための同数の独立した命令フィールドを持つ１個の長い命令語ＶＬＩＷであって、その一部に解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を有し、次に解釈・実行すべき命令語ＶＬＩＷを保持するための命令レジスタと、

この命令レジスタに保持されている前記命令語ＶＬＩＷの各命令フィールドに対応してそれぞれ設けられ、対応する命令フィールドの内容を解釈して当該フィールドに対応する前記演算器を制御するデコード手段であって、当該フィールドが前記命令有効化情報により無効命令フィールドであることが示されている場合には、当該フィールドの内容に従う対応する前記演算器の動作を禁止するデコード手段と、

前記命令有効化情報により無効であることが示されている前記命令語ＶＬＩＷ上の命令フィールドに対して、演算オペランドとしてアクセスすることを許すメモリ管理手段とを具備することを特徴とするＶＬＩＷ方式の計算機システム。

【請求項２】　ソースプログラムをコンパイルして前記計算機システム内で実行される命令語ＶＬＩＷの列からなるオブジェクトプログラムを生成するコンパイル手段であって、前記命令語ＶＬＩＷ中の解釈・実行が不要な無効命令フィールドを演算オペランド領域として、当該フィールドに定数または変数を割り当てるコンパイル手段を更に具備することを特徴とする請求項１記載のＶＬＩＷ方式の計算機システム。

【請求項３】　並列動作可能な複数の演算器を同数の独立した命令フィールドを持つ１個の長い命令語ＶＬＩＷにより制御するＶＬＩＷ方式の計算機システムにおいて、

前記命令語ＶＬＩＷの列からなるオブジェクト群をリンクしてロードモジュールを生成するリンク手段であって、当該ロードモジュールの生成時に、前記演算器を動作させる有効命令フィールドを除く無効命令フィールド部分を削除するリンク手段と、

前記ロードモジュールから指定ページの命令語ＶＬＩＷの列を主記憶上にロードするためのデマンドページアクセス要求が発生した場合に、前記無効命令フィールド部分を含む命令語ＶＬＩＷの列を復元するデマンドページアクセス手段とを具備することを特徴とするＶＬＩＷ方式の計算機システム。

【請求項４】　前記命令語ＶＬＩＷは、その一部に解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を有しており、

前記リンク手段は、前記各命令語ＶＬＩＷ毎に、そのＶＬＩＷの命令有効化情報をもとに前記無効命令フィールドの位置を確認して、当該フィールドを取り除き、

2

前記デマンドページアクセス手段は、前記ロードモジュールから指定ページの命令語ＶＬＩＷの列を主記憶上にロードする際に、対応する前記命令語ＶＬＩＷの命令有効化情報を参照して、当該ＶＬＩＷ上の前記無効命令フィールドの位置を検出し、その位置を飛ばして当該ＶＬＩＷ上の有効命令フィールドの内容を前記主記憶に配置することを特徴とする請求項３記載のＶＬＩＷ方式の計算機システム。

【請求項５】　前記リンク手段は、前記リンク時に、リンク前の前記命令語ＶＬＩＷの列における前記有効命令フィールドと無効命令フィールドの配置を示す命令マップを生成して前記ロードモジュールに付加し、

前記デマンドページアクセス手段は、前記ロードモジュールから指定ページの命令語ＶＬＩＷの列を主記憶上にロードする際に、当該ロードモジュールに付されている前記命令マップに従って、前記無効命令フィールドの位置を検出し、その位置にノーオペレーション命令ｎｏｐを配置することを特徴とする請求項３記載のＶＬＩＷ方式の計算機システム。

【請求項６】　並列動作可能な複数の演算器を同数の独立した命令フィールドを持つ１個の長い命令語ＶＬＩＷにより制御するＶＬＩＷ方式の計算機システムに適用されるＶＬＩＷの解釈・実行方法において、

前記命令語ＶＬＩＷの一部に、解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を設け、

解釈・実行すべき前記命令語ＶＬＩＷ上に、当該ＶＬＩＷの前記命令有効化情報により無効であることが示されている命令フィールドが存在する場合には、その命令フィールドの内容に従う対応する前記演算器の動作を禁止し、

前記命令有効化情報により無効であることが示されている前記ＶＬＩＷ上の命令フィールドに対して、演算オペランドとしてアクセスすることを許すことにより、

当該フィールドの演算オペランド領域としての利用を許すようにしたことを特徴とするＶＬＩＷの解釈・実行方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、並列動作可能な複数の演算器を独立した複数の命令（命令フィールド）の集まりであるＶＬＩＷ（Very Long Instruction Word）と称される１個の長い命令語により制御するＶＬＩＷ方式の計算機システム及びＶＬＩＷの解釈・実行方法に関する。

【０００２】

【従来の技術】近年、処理の高速化のために、ＶＬＩＷ方式の計算機システムが開発されている。この種の計算機システムは、並列動作可能な複数の演算器を備えており、ＶＬＩＷ上の独立した各命令により、当該各演算器

を同期をとって個々に且つ並列に制御するようになっている。

【０００３】図１３は、ＶＬＩＷ方式の計算機システムの要部の基本構成を示すものである。この図１３のシステムでは、命令＃１～＃４の４命令を１組とするＶＬＩＷ１３１により、４つの演算器１３２-1～１３２-4が制御される構成となっている。即ち、図１３のシステムにおいて、ＶＬＩＷ１３１中の各命令＃１～＃４は、それぞれデコーダ（ＤＥＣ）１３３-1～１３３-4により同時に且つ並列に解釈され、それぞれ対応する演算器１３２-1～１３２-4の動作を制御する。

【０００４】このように、ＶＬＩＷ方式の計算機システムでは、ＶＬＩＷにより複数の演算器を並列動作させることができる。したがって、プログラマまたはコンパイラが、並列に実行可能な演算を調べ、これらの実行をＶＬＩＷ上の対応する命令にプログラミングすることで、命令語レベルの並列処理が実現され、処理の高速化が図れる。

【０００５】さて、この種の計算機システムで使用されるＶＬＩＷ方式のプログラム（オブジェクトプログラム）では、実行する計算アルゴリズムの因果関係（各命令間の依存関係）によっては、本来並列動作可能な複数の演算器の全てを演算に使用することができないケースが発生する。

【０００６】このような場合、使用しない演算器に対応するＶＬＩＷ中の命令フィールドには、演算を行わないことを明示するノーオペレーション命令（ｎｏｐ命令）が配置される。このｎｏｐ命令の配置により、誤動作防止が図られる。

【０００７】

【発明が解決しようとする課題】上記したように、ＶＬＩＷ方式の計算機システムでは、各命令間の依存関係により同時に実行できる命令が所定数揃わない場合、ＶＬＩＷ中の一部の命令フィールドに誤動作防止のためにｎｏｐ命令が配置されるようになっていた。

【０００８】このため、演算器の数を増やし、それに合わせてＶＬＩＷの命令語長を極端に長くした場合や、もともと並列動作しにくい処理をプログラミングした場合などには、実行コードの大半がｎｏｐ命令となってしまい。オブジェクトプログラムサイズ（実行オブジェクトサイズ）が著しく大きくなって、そのプログラムの占める主記憶領域も大きくなるという問題があった。このことは、ディスク（補助記憶）上の実行ロードモジュールについても同様であった。

【０００９】この問題を解決するのに、ＶＬＩＷの命令形式を可変長としてｎｏｐ命令の部分を削除することで、コードサイズを削減し、ＶＬＩＷを解釈（デコード）・実行する際に、ハードウェア的に本来の形式に復元する方式が考えられる。しかし、この方式はハードウェア構成が極めて複雑になり、また性能も悪くなるため、実用的でない。しかも、分岐先の計算など、プログラムの制御構造の実現が極めて複雑になる。

【００１０】本発明は上記事情を考慮してなされたものでその目的は、従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができるＶＬＩＷ方式の計算機システム及びＶＬＩＷの解釈・実行方法を提供することにある。

【００１１】本発明の他の目的は、従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分を変数または定数領域として利用でき、もって実行オブジェクトが使用する主記憶領域を削減できるようにすることにある。本発明の更に他の目的は、ロードモジュールファイルのサイズを小さくできるようにすることにある。

【００１２】

【課題を解決するための手段及び作用】本発明の第１の観点に係る構成は、並列動作可能な複数の演算器と、これら各演算器を個々に制御するための同数の独立した命令フィールドを持つＶＬＩＷであって、その一部に解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を有し、次に解釈・実行すべきＶＬＩＷを保持するための命令レジスタと、この命令レジスタに保持されているＶＬＩＷの各命令フィールドに対応してそれぞれ設けられ、対応する命令フィールドの内容を解釈して当該フィールドに対応する演算器を制御するデコード手段であって、当該フィールドが上記命令有効化情報により無効命令フィールドであることが示されている場合には、当該フィールドの内容に従う対応する演算器の動作を禁止するデコード手段と、上記命令有効化情報により無効であることが示されているＶＬＩＷ上の命令フィールドに対して、演算オペランドとしてアクセスすることを許すメモリ管理手段とを備えたことを特徴とするものである。

【００１３】上記第１の観点に係る構成において、命令レジスタに次に実行すべきＶＬＩＷがフェッチされた場合には、デコード手段は、そのＶＬＩＷの一部をなす命令有効化情報により有効命令フィールドであることが示されているフィールドの内容については、解釈（デコード）を行って、その解釈結果に従って対応する演算器の動作を制御する。また、命令有効化情報により無効命令フィールドであることが示されているフィールドの内容については、解釈を行わず、対応する演算器を動作させない。なお、解釈自体は行うものの、その解釈結果を無視し、対応する演算器を動作させないようにしても構わない。

【００１４】このように、ＶＬＩＷ中の無効命令フィールドに対応する演算器は、当該ＶＬＩＷの命令有効化情報に基づくデコード手段の制御により、その無効命令フィールドの内容に従って動作することが禁止される。このため、その無効命令フィールドに従来とは異なってｎｏｐ命令を配置せずに、その無効命令フィールドを演算

オペランドの領域として用いるようにしても、誤動作を招くことはない。この場合、従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができる。ここでＶＬＩＷ中の無効命令フィールドを演算オペランドの領域として利用するには、ソースプログラムをコンパイルしてＶＬＩＷの列からなるオブジェクトプログラムを生成する際に、ＶＬＩＷ中の解釈・実行が不要な無効命令フィールドに定数または変数を割り当てるようにすればよい。

【００１５】本発明の第２の観点に係る構成は、並列動作可能な複数の演算器を同数の独立した命令フィールドを持つＶＬＩＷにより制御するＶＬＩＷ方式の計算機システムにおいて、ＶＬＩＷの列からなるオブジェクト群をリンクしてロードモジュールを生成するリンク手段であって、当該ロードモジュールの生成時に、上記演算器を動作させる有効命令フィールドを除く無効命令フィールド部分を削除するリンク手段と、上記ロードモジュールから指定ページのＶＬＩＷの列を主記憶上にロードするためのデマンドページアクセス要求が発生した場合に、上記無効命令フィールド部分を含むＶＬＩＷの列を復元するデマンドページアクセス手段とを備えたことを特徴とするものである。

【００１６】上記第２の観点に係る構成において、リンク手段によるリンク対象となるオブジェクト群を構成する各ＶＬＩＷには、従来とは異なって、その一部に、解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報が設けられている。

【００１７】リンク手段は、各ＶＬＩＷ毎に、そのＶＬＩＷの命令有効化情報をもとに無効命令フィールドの位置を確認して、当該フィールドを取り除くことにより、無効命令フィールドが削除されたロードモジュールを生成する。これにより、ロードモジュールが置かれるファイルのサイズを削減することが可能となる。

【００１８】一方、デマンドページアクセス手段は、デマンドページアクセス要求が発生すると、上記ロードモジュールから指定ページのＶＬＩＷの列を主記憶上にロードする。但し、当該ロードモジュールには無効命令フィールド部分は含まれていない。

【００１９】そこでデマンドページアクセス手段は、指定ページのＶＬＩＷの列を主記憶上にロードする際には、対応するＶＬＩＷの命令有効化情報を参照して、当該ＶＬＩＷ上の前記無効命令フィールドの位置を検出し、その位置を飛ばして当該ＶＬＩＷ上の有効命令フィールドの内容を主記憶に配置する。これにより、ＶＬＩＷが主記憶上に復元・配置される。

【００２０】この他、命令有効化情報を持たない従来と同様の形式のＶＬＩＷ、即ち無効命令フィールドにｎｏｐ命令が設定されるＶＬＩＷについても、無効命令フィールド（ｎｏｐ命令）の削除によるロードモジュールのサイズの削減と、デマンドページアクセス時のＶＬＩＷの復元を実現することが可能である。

【００２１】このためには、まずリンク手段によるリンク時には、各ＶＬＩＷ毎に、そのＶＬＩＷ中のｎｏｐ命令を検出して、そのｎｏｐ命令を除くことにより、ｎｏｐ命令が削除されたロードモジュールを生成すればよい。また、デマンドページアクセス時のＶＬＩＷの復元のために、リンク手段によるリンク時に、ＶＬＩＷの列におけるｎｏｐ命令（が設定される無効命令フィールド）とそれ以外の命令（が設定される有効命令フィールド）の配置を示す命令マップを生成してロードモジュールに付加しておく。

【００２２】この場合、デマンドページアクセス手段は、ロードモジュールから指定ページのＶＬＩＷの列を主記憶上にロードする際に、当該ロードモジュールに付されている命令マップに従って、ｎｏｐ命令の位置を検出し、その位置にｎｏｐ命令を配置することで、元のＶＬＩＷを主記憶上に復元・配置できる。

【００２３】

【実施例】以下、本発明の実施例につき図面を参照して説明する。

［第１の実施例］図１は本発明の第１の実施例に係るＶＬＩＷ方式の計算機システムの構成を示すブロック図、図２は同実施例で適用されるＶＬＩＷの命令フォーマットを示す図である。

【００２４】まず、本実施例で適用されるＶＬＩＷの命令語長は１２８ビットである。この１２８ビットのＶＬＩＷは、図２に示すように、４つの３２ビットの命令フィールドに分割される。

【００２５】ＶＬＩＷの１番目の命令フィールド（図面上で最も左側の命令フィールド）のうちの上位３ビット（図面上で最も左側の３ビット）はビットａ２～ａ４からなる命令有効化情報ａの設定フィールド（命令有効化フィールド）として用いられ、残りの２９ビットはＶＬＩＷの１番目の命令（命令＃１）を配置するのに用いられる。また、ＶＬＩＷの２番目～４番目の命令フィールドは、ＶＬＩＷの２番目～４番目の命令（命令＃２～＃４）を配置するのに用いられる。

【００２６】命令有効化情報ａを構成するビットａ２～ａ４は、ＶＬＩＷ中のそれぞれ２番目～４番目の命令フィールドの設定内容が解釈・実行の必要のある有効な命令である（“１”の場合）か否か（“０”の場合）を示す。本実施例において、命令有効化情報ａのビットａｉ（ｉ＝２～４）が“０”の場合、ＶＬＩＷ中のｉ番目の命令フィールドは、変数または定数が設定される演算オペランド領域（以下、変数／定数領域と称する）として利用可能である。

【００２７】図１の計算機システムにおいて、１０はシステムの中枢をなすＣＰＵ、２０はＶＬＩＷ方式のプログラム（オブジェクトプログラム）、オペランド等が格納される主記憶である。

【００２８】ＣＰＵ１０は、主記憶２０上の命令語群（ＶＬＩＷ群）の一部の写しが置かれる命令キャッシュ１１と、この命令キャッシュ１１からフェッチされたＶＬＩＷが保持される命令レジスタ１２と、この命令レジスタ１２に保持されたＶＬＩＷ中の１番目～４番目の命令フィールドの設定内容の解釈（デコード）を行うデコーダ（ＤＥＣ）１３-1～１３-4とを備えている。

【００２９】デコーダ１３-1は、ＶＬＩＷ中の１番目の命令フィールドの（命令有効化情報ａを除く部分の）設定内容の解釈を無条件で行うものである。一方、他のデコーダ１３-2～１３-4は、ＶＬＩＷ中の２番目～４番目の命令フィールドの設定内容の解釈を、ＶＬＩＷ中の命令有効化情報ａのビットａ2 ～ａ4 の状態（“1”であるか否か）に応じて行うものである。

【００３０】ＣＰＵ１０はまた、デコーダ１３-1～１３-4の解釈結果に応じて演算を行う演算器１４-1～１４-4を持ち、命令実行と実行結果の書き込みを司る実行・書き込み部１５と、主記憶２０上のオペランド群の一部の写しが置かれるオペランドキャッシュ１６とを備えている。このオペランドキャッシュ１６のキャッシュライン長はＶＬＩＷ長と同じ（ここでは１２８ビット）であるものとする。そして、一部の命令フィールドが変数／定数領域として利用されているＶＬＩＷがオペランドデータとして当該オペランドキャッシュ１６に保持される場合、そのＶＬＩＷは、複数のキャッシュラインにまたがることなく、１２８ビットアラインに配置されるるものとする。

【００３１】ＣＰＵ１０は更に、主記憶２０のページ管理、（論理アドレスから物理アドレスへの）アドレス変換等を司るメモリ管理部１７を備えている。メモリ管理部１７は、論理アドレスを物理アドレスに高速に変換するためのアドレス変換バッファ機構であるＴＬＢ（translation lookaside buffer）１７１と、アクセス権の制御等を行うアクセス制御回路１７２とを備えている。

【００３２】ＴＬＢ１７１は、メモリ保護のためのアクセス権情報、論理アドレス（ここでは論理ページアドレス）をもとに生成されるタグ情報（アドレスタグ）、及び当該論理アドレスに対応する物理アドレス（ここでは物理ページアドレス）を含む情報を保持するための複数のエントリを有している。アクセス権情報は、対応するページに対してリードのみ許可するアクセス権ＲＯ、リードとライトの両方を許可するアクセス権ＲＷ及びアクセス権Ｅ等のいずれかを示すものである。アクセス権Ｅは、対応するページがＶＬＩＷの格納領域として用いられており、アクセス先となるＶＬＩＷ中の領域に有効な命令がないならば、その領域を通常のオペランドアクセスと同様にアクセスできるという“実行可”を示す。

【００３３】主記憶２０は、メモリ管理部１７によりページ単位で管理される。主記憶２０内の領域は、例えばＶＬＩＷが格納されるページと、オペランドデータが格納されるページ等に分けられる。但し、本実施例においては、後述するようにオペランドを含むＶＬＩＷが存在する。

【００３４】次に、本発明の第１の実施例の動作を説明する。まず、ＣＰＵ１０において命令フェッチ要求が発生したものとする。もし、要求した命令語（ＶＬＩＷ）が命令キャッシュ１１に存在するキャッシュヒット時には、その命令語（ＶＬＩＷ）が命令キャッシュ１１から命令レジスタ１２に高速に読み込まれる。

【００３５】一方、要求した命令語（ＶＬＩＷ）が命令キャッシュ１１に存在しないミスヒット時には、その命令語（ＶＬＩＷ）はメモリ管理部１７を通して主記憶２０からフェッチされ、命令キャッシュ１１に保持された後、当該命令キャッシュ１１から命令レジスタ１２にフェッチされる。

【００３６】なお、キャッシュヒット／ミスヒットの判定アルゴリズムや、命令キャッシュ１１への命令語（ＶＬＩＷ）の登録処理等については、従来からよく知られており、しかも本発明に直接関係しないため説明を省略する。

【００３７】命令レジスタ１２に読み込まれたＶＬＩＷ中の（１番目の命令フィールドの上位部分に設定された）３ビット命令有効化情報ａのビットａ2 ～ａ4 は、デコーダ１３-2～１３-4に導かれる。また、当該ＶＬＩＷ中の１番目の命令フィールドの（命令有効化情報ａを除く部分の）設定内容（命令＃１）はデコーダ１３-1に導かれ、２番目の命令フィールドの設定内容（命令＃２または変数／定数）はデコーダ１３-2に導かれる。更に、当該ＶＬＩＷ中の３番目の命令フィールドの設定内容（命令＃３または変数／定数）はデコーダ１３-3に導かれ、４番目の命令フィールドの設定内容（命令＃４または変数／定数）はデコーダ１３-4に導かれる。

【００３８】デコーダ13-1は、命令レジスタ１２から導かれたＶＬＩＷの１番目の命令フィールドの設定内容（命令＃１）を無条件で解釈（デコード）し、その解釈結果に応じて実行・書き込み部１５の演算器１４-1を制御する。

【００３９】一方、デコーダ１３-2～１３-4は、命令レジスタ１２から導かれた命令有効化情報ａのビットａ2 ～ａ4 に応じて、次のように動作する。まず、命令有効化情報ａのビットａ2 ～ａ4 が“1”の場合には、デコーダ１３-2～１３-4は、命令レジスタ１２から導かれたＶＬＩＷの２番目～４番目の命令フィールドに解釈・実行の必要な有効な命令（命令＃２～＃４）が設定されているものとして、その命令フィールドの設定内容（命令＃２～＃４）を解釈（デコード）し、その解釈結果に応じて実行・書き込み部１５の演算器１４-2～１４-4を制御する。

【００４０】次に、命令有効化情報ａのビットａ2 ～ａ4 が“0”の場合には、デコーダ１３-2～１３-4は、命

令レジスタ１２から導かれたＶＬＩＷの２番目～４番目の命令フィールドには解釈・実行の必要な有効な命令は設定されておらず、代わりに変数／定数が設定されている可能性があるものとして、その命令フィールドを無効扱い（無効命令フィールド、不要命令フィールド）とし、その命令フィールドの設定内容の解釈を控えて、演算器１４-2～１４-4を動作させない。なお、解釈（デコード）動作自体は行っても、その解釈結果を無視することで、対応する演算機を動作させないようにしても構わない。

【００４１】以上のデコーダ１３-2～１３-4の動作により、ＶＬＩＷの一部（ここでは１番目の命令フィールドの先頭部分）に設けられた命令有効化情報ａのビットａ２～ａ４で命令の無効化が指定されているならば、ＶＬＩＷの対応する２番目～４番目の命令フィールドには、どのような３２ヒットデータを配置しても、そのフィールドに対応する演算器１４-2～１４-4が誤動作することはない。

【００４２】したがって、従来であればｎｏｐ命令を配置する必要のあるＶＬＩＷ中の命令フィールドを変数または定数の設定領域（変数／定数領域）として利用することができる。但し、１番目の命令フィールドの設定内容は、デコーダ１３-1により常時解釈されて、実行されるため、実行が不要な場合には、当該フィールドに従来通りｎｏｐ命令を配置する必要がある。

【００４３】なお、本実施例においては、１つのＶＬＩＷ中に配置される命令の長さが、２９ビットと３２ビットの２種類ある（図２参照）。しかし、ＶＬＩＷ自体は固定長であり、しかも２９ビット長の命令（命令＃１）は常にＶＬＩＷの１番目の命令フィールドに設定され、３２ビット長の命令（命令＃２～＃４）は常にＶＬＩＷの２番目～４番目の命令フィールドに設定されることから、何ら問題とはならず、ハードウェア構成も（ｎｏｐ命令を削除してＶＬＩＷを可変長とする方式とは異なって）図１に示したように簡単である。また、ＶＬＩＷの１番目の命令フィールドに対応する演算器１４-1で実行可能な命令の種類は、他の命令フィールドに対応する演算器１４-2～１４-4で実行可能な命令の種類より少なくなるが、演算器１４-1で実行できない命令をＶＬＩＷの２番目以降の命令フィールドに配置して演算器１４-2～１４-4で実行させることで対処できる。

【００４４】さて、デコーダ１３-1～１３-4の少なくとも１つで命令の解釈が行われて、実行・書き込み部１５内の対応する演算器が動作する際に、その動作に必要なオペランド（変数／定数）を（例えばオペランドキャッシュ１６から）リードする必要がある、或いは演算器の動作が行われた後、その演算結果を（例えばオペランドキャッシュ１６に）ライトする必要があるものとする。この場合、実行・書き込み部１５からメモリ管理部１７に対してオペランドアクセス（オペランドデータアクセス）要求が発生する。

【００４５】するとメモリ管理部１７では、図３のフローチャートに従って以下に述べるオペランドアクセス処理が行われる。まず、メモリ管理部１７内のアクセス制御回路１７２は、実行・書き込み部１５からのオペランドアクセス要求を受け取ると、ＴＬＢ１７１を参照し、その要求先のアドレス（論理ページアドレス）に対応する物理ページアドレスが登録されているエントリを検索する。

【００４６】もし、ＴＬＢ１７１に目的とするエントリが存在するヒット時であれば、アクセス制御回路１７２は当該エントリ中の物理ページアドレスを用いて論理アドレスから物理アドレスへの周知のアドレス変換を高速に行うと共に、当該エントリ中のアクセス権情報を得る。

【００４７】一方、ＴＬＢ１７１に目的とするエントリが存在しないミスヒット時であれば、アクセス制御回路１７２は、主記憶２０の所定領域に置かれているページテーブル（図示せず）をアクセスし、当該ページテーブルを用いて論理アドレスから物理アドレスへの周知のアドレス変換を行う。このページテーブルには、ＯＳ（オペレーティングシステム）の管理により、各ページ毎のアクセス権情報が設定されている。このアクセス権情報としては、リードのみ許可するアクセス権ＲＯ、リードとライトの両方を許可するアクセス権ＲＷなど従来からよく知られているメモリ保護のためのアクセス権の他に、本実施例特有のアクセス権Ｅが用意されている。このアクセス権Ｅは、前記したように、対応するページがＶＬＩＷの格納領域として用いられており、アクセス先となるＶＬＩＷ中の領域に有効な命令がないならば、その領域を通常のオペランドアクセスと同様にアクセスできるという“実行可”を示すものである。本実施例において、ＲＯ＝０，ＲＷ＝１，Ｅ＝２である。

【００４８】アクセス制御回路１７２は、ページテーブルを用いたアドレス変換を行うと、当該テーブルから得られる該当ページのアクセス権情報、要求先の論理アドレス（ここでは論理ページアドレス）をもとに生成されるタグ情報（アドレスタグ）、及びアドレス変換結果である物理ページアドレスを含む情報をＴＬＢ１７１内のエントリに登録する周知の登録動作を行う。

【００４９】さてアクセス制御回路１７２は、ＴＬＢ１７１またはページテーブルを用いたアドレス変換により、要求先アドレス（論理アドレス）に対応する物理アドレスとアクセス権情報を得ると（ステップＳ１）、オペランドのリード時であれば、アクセス権情報の示すアクセス権がＲＯ或いはＲＷの場合、オペランドのライト時であればアクセス権情報の示すアクセス権がＲＷの場合に、取得した物理アドレスをもとにオペランドキャッシュ１６を対象とするオペランドアクセスを実行し、リード時であればリードしたオペランドを実行・書き込み

部１５に渡す。この場合の動作は、従来と何ら変わらない。なお、オペランドキャッシュ１６に目的とするデータが存在しないミスヒット時には、主記憶２０がアクセスされることはいうまでもない。

【００５０】これに対し、アクセス権情報の示すアクセス権がＥの場合、即ち実行可の場合には（ステップＳ２）、アクセス制御回路１７２は、取得した物理アドレスをもとにオペランドキャッシュ１６の該当するキャッシュラインをリードアクセスする（ステップＳ３）。アクセス権がＥ（実行可）の場合、アクセスしたキャッシュラインの情報はＶＬＩＷである。

【００５１】アクセス制御回路１７２は、アクセスしたキャッシュラインの情報（ＶＬＩＷ）の先頭の３２ビット、即ちＶＬＩＷの１番目の命令フィールドから命令有効化情報ａを取り出す（ステップＳ４）。なお、ミスヒット時には、命令有効化情報ａの取り出しは、主記憶２０を対象に行われる。

【００５２】アクセス制御回路１７２は、命令有効化情報ａの取り出しを行うと、オペランドアクセスすべきフィールド（データ語部分）に有効な命令が存在するか否かを、当該命令有効化情報ａの対応ビット（の状態）をもとに調べる（ステップＳ５）。

【００５３】もし、有効な命令が存在するならば、アクセス制御回路１７２は、不正アクセス（アクセスエラー）として、要求されたオペランドアクセスを行わずに当該命令を保護し、オペランドキャッシュ１６にその旨を通知する（ステップＳ６）。

【００５４】これに対し、有効な命令が存在しないならば、アクセス制御回路１７２は、通常のオペランドデータと同様に、オペランドアクセスすべきフィールド（データ語部分）を対象とするアクセスを行う（ステップＳ７）。

【００５５】このように、ＶＬＩＷ中２番目～４番目の命令フィールドのうち、命令の依存関係で従来であればｎｏｐ命令を配置しなければならない命令フィールドを変数／定数領域として利用することで、従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができる。しかも、変数／定数領域として利用する命令フィールドの設定内容は、命令有効化情報ａの対応ビットの無効化指定により、対応するデコーダ１３-ｉ（ｉは２～４のいずれか）で解釈されず、無効化扱いされるため、演算器１４-ｉが誤動作する虞はない。

【００５６】図４は図１の計算機システムで適用される、図２に示したような形式のＶＬＩＷ群を含むオブジェクトプログラムをソースプログラムから生成するコンパイラの構成を示す。

【００５７】図４のコンパイラは、ソースプログラム３１をコンパイルして従来と同様の形式のＶＬＩＷ群を含む第１のオブジェクトプログラム３２を生成する第１のコンパイル部４１と、この第１のオブジェクトプログラム３２から図２の形式の第２のオブジェクトプログラム３３を生成する第２のコンパイル部４２とから構成される。

【００５８】次に、図４の構成のコンパイラの動作を、図５及び図６のフローチャートを参照して説明する。まず、第１のコンパイル部４１は、ソースプログラム３１のソースコード列から、各ソースコード毎に、逐次実行型の命令または命令群を作成し、ＶＬＩＷへの再配置前の命令列（オブジェクトコード列）を生成する（ステップＳ１１）。

【００５９】次に第１のコンパイル部４１は、生成した再配置前の命令列を４つの３２ビット命令フィールドを持つＶＬＩＷ型に最適化、再配置し、各命令間の依存関係により演算器を動作させる命令を配置できない命令フィールドにはｎｏｐ命令を配置することで、従来と同様の形式のＶＬＩＷ群を含む第１のオブジェクトプログラム３２を生成する（ステップＳ１２）。但し、ＶＬＩＷの１番目の命令フィールドには、先頭の３ビットを除く領域に２９ビット命令またはｎｏｐ命令が配置される点で、従来と少し異なる。なお、２９ビット命令は、上位の３ビットに無関係に演算指定可能な３２ビット命令と等価である。

【００６０】上記のようにして第１のコンパイル部４１により生成される第１のオブジェクトプログラム３２は、（従来と同様の形式の）ＶＬＩＷ群からなるコード部３２１と、コード部３２１上の命令から参照されるデータ（変数／定数）の集合であるデータ部３２２から構成される。このデータ部３２２のデータは、そのデータが置かれる論理的な位置を示す情報（ラベル）と関連付けられている。

【００６１】第１のコンパイル部４１により第１のオブジェクトプログラム３２が生成されると、第２のコンパイル部４２が起動される。すると第２のコンパイル部４２は、第１のオブジェクトプログラム３２のコード部３２１の先頭ＶＬＩＷの２番目の命令フィールドを参照して（ステップＳ１３）、そこにｎｏｐ命令が配置されているか否かをチェックする（ステップＳ１４）。

【００６２】第２のコンパイル部４２は、ｎｏｐ命令が配置されている命令フィールドを検出した場合、そのフィールドを変数／定数領域として利用するならば（ステップＳ１５）、そのフィールドに第１のオブジェクトプログラム３２のデータ部３２２上の任意のデータを（ｎｏｐ命令に代えて）配置する（ステップＳ１６）。

【００６３】次に第２のコンパイル部４２は、コード部３２１をスキャンして、配置したデータを参照する命令を全て探し、これら各命令に設定されている当該データの参照先を示すラベルを、ステップＳ１６で配置した命令フィールドの位置を示すラベルにそれぞれ書き換える（ステップＳ１７）。なお、第１のコンパイル部４１に

より第１のオブジェクトプログラム３２を生成する際に、コード部３２１上の各命令のうち、データ部３２２上のデータを参照する命令の位置情報を、当該データと対応させてテーブル等に保持しておくならば、上記のステップＳ１７でのラベル書き換えの対象となる命令を簡単に探すことができる。

【００６４】第２のコンパイル部４２は、ステップＳ１７を実行すると、現在注目しているＶＬＩＷ（現ＶＬＩＷ）の最後の命令フィールドまで終了したか否かをチェックする（ステップＳ１８）。このステップＳ１８のチェックは、参照した命令フィールドにｎｏｐ命令が配置されていない場合と、参照した命令フィールドを変数／定数領域として利用しない場合にも行われる。

【００６５】もし、現ＶＬＩＷの最後の命令フィールドまで終了していなければ、第２のコンパイル部４２は、現ＶＬＩＷの次の命令フィールドを参照して（ステップＳ１９）、ステップＳ１４に戻る。

【００６６】一方、現ＶＬＩＷの最後の命令フィールドまで終了しているならば、第２のコンパイル部４２は、そのＶＬＩＷの１番目の命令フィールドの上位３ビット部分に、２番目～４番目の命令フィールドの内容に応じた命令有効化情報ａを設定する（ステップＳ２０）。

【００６７】第２のコンパイル部４２は、命令有効化情報ａの設定処理を行うと、コード部３２１の最後のＶＬＩＷまで終了したか否かをチェックする（ステップＳ２１）。

【００６８】もし、コード部３２１の最後のＶＬＩＷまで終了していなければ、第２のコンパイル部４２は、コード部３２１上の次のＶＬＩＷの２番目の命令フィールドを参照して（ステップＳ２２）、ステップＳ１４に戻る。

【００６９】一方、コード部３２１の最後のＶＬＩＷまで終了しているならば、第２のコンパイル部４２はコンパイル処理を終了する。これにより、図２の形式のＶＬＩＷの群からなるコード部３３と、コード部３２１上の命令から参照されるデータ（変数／定数）の集合であるデータ部３２２から構成される、第２のオブジェクトプログラム３３が生成される。この第２のオブジェクトプログラム３３が、図１の計算機システムで用いられる。

【００７０】以上、従来であればｎｏｐ命令を配置する必要のあるＶＬＩＷ中の命令フィールドを変数／定数領域として利用できるようにすることで、実行オブジェクトが使用する主記憶２０の領域の削減を図った第１の実施例につき説明した。

【００７１】なお、上記第１の実施例では、１２８ビット長のＶＬＩＷの１番目の３２ビット長命令フィールドの一部を命令有効化情報ａとして用いているが、４つの３２ビット長の命令フィールドと命令有効化情報ａからなる１３１ビット長のＶＬＩＷを用いるようにしても構わない。いずれの場合も、ＶＬＩＷの一部を命令有効化情報ａとして用いることに変わりはない。また、命令フィールドの長さ、命令フィールドの数も、上記第１の実施例に限るものでないことは勿論である。

【００７２】また、上記第１の実施例では、図４のフローチャートに従うコンパイル処理のステップＳ１２において、従来と同様の形式のＶＬＩＷの列を含む第１のオブジェクトプログラム３２を生成し、命令有効化情報ａの設定はステップＳ２０で行うものとしたが、ステップＳ１２の段階で、無効命令フィールドにｎｏｐ命令とは別の無意味なコードを配置するとか、或いは当該フィールドを空きフィールドとし、ＶＬＩＷの先頭命令フィールドに命令有効化情報ａを配置するようにしてもよい。

［第２の実施例］次に、ＶＬＩＷ方式のロードモジュールファイルのサイズを小さくできるようにすることで、当該ファイルの占める補助記憶領域（ディスク領域）の削減を図るようにした第２の実施例につき、図面を参照して説明する。

【００７３】図７はロードモジュールを生成して補助記憶に格納しておき、テマンドページアクセス要求が発生した場合に、要求されたロードモジュールのページを主記憶に展開するための構成を示す。

【００７４】同図において、５１はリンク手段としてのリンケージエディタ、５２はオペレーティングシステム（ＯＳ）の仮想記憶管理部、５３は主記憶である。リンケージエディタ５１は、ＶＬＩＷ方式のオブジェクト群（オブジェクトプログラム群）６１をリンクして計算機システム（内のＣＰＵ）で実行可能な圧縮されたロードモジュール６２を生成するものである。本実施例で適用されるオブジェクト群６１は、前記第１の実施例で適用した図２の形式のＶＬＩＷの列、即ち４つの４バイト命令フィールドから構成され（ＶＬＩＷ長＝１６バイト）、１番目の命令フィールドの先頭部分に命令有効化情報ａを持つＶＬＩＷの列からなるものとする。但し、命令有効化情報ａ内のビットａｉ（ｉ＝２～４）により（解釈・実行の必要がないものとして）無効（不要）指定される命令フィールドは、前記第１の実施例とは異なって、変数／定数領域として用いられず空き領域として位置付けられるものとする。

【００７５】ロードモジュール６２は、補助記憶、例えばディスク装置に置かれるロードモジュールファイル６３に格納されるもので、ページ数分のコード部６２１と、ページ数分のインデックス部６２２とから構成される。

【００７６】各コード部６２１は、対応するページのＶＬＩＷ列を圧縮した命令群（具体的にはＶＬＩＷ列から無効命令フィールド部分が削除された残りの命令群）が格納される圧縮コード領域として用いられるものである。

【００７７】各インデックス部６２２は、対応するページのコード部６２１の先頭位置を（ロードモジュールフ

ファイル６３内の相対位置で）示す情報（以下、ページインデックスと称する）を格納しておくためのものである。

【００７８】仮想記憶管理部５２は、デマンドページアクセス要求が発生した場合に、ロードモジュールファイル６３内のロードモジュール６２から対応するページのＶＬＩＷ列を復元して主記憶５３に配置するものである。

【００７９】次に、図７の構成の動作を、（１）リンク処理と（２）デマンドページアクセス処理に分けて、図８乃至図１０のフローチャートを参照して順に説明する。

（１）リンク処理

まず、リンケージエディタ５１によるリンク処理について説明する。

【００８０】リンケージエディタ５１は、オブジェクト群６１をリンクしてロードモジュール６２を生成する際には、必要とするページ数分のインデックス部６２２をロードモジュールファイル６２内に確保する（ステップＳ３１）。

【００８１】次にリンケージエディタ５１は、ロードモジュール６２の先頭ページ用のコード部６２１をロードモジュールファイル６２内に確保し、そのコード部６２１の先頭のロードモジュールファイル６３内相対位置を示すページインデックスを、対象ページに固有のインデックス部６２２に設定する（ステップＳ３２）。

【００８２】次にリンケージエディタ５１は、オブジェクト群６１から対象ページの先頭ＶＬＩＷを取り出す（ステップＳ３３）。そしてリンケージエディタ５１は、取り出したＶＬＩＷの１番目の命令フィールドの先頭部分に設定されている命令有効化情報ａを参照し、解釈・実行の必要のない無効（不要）フィールドであることが示されている命令フィールド部分を当該ＶＬＩＷから取り除いて、コード部６２１に例えば１命令フィールド単位で格納する（ステップＳ３４）。

【００８３】これにより、命令有効化情報ａ（中のビットａ２～ａ４）が例えば“０１０”であるＶＬＩＷの場合には、当該ＶＬＩＷ中の２番目と４番目の命令フィールド部分が削除され、コード部６２１には、残りの１番目と３番目の命令フィールドの内容だけが格納される。同様に、命令有効化情報ａ（中のビットａ２～ａ４）が例えば“１０１”であるＶＬＩＷの場合には、当該ＶＬＩＷ中の３番目の命令フィールド部分が削除され、コード部６２１には、残りの１番目と２番目と４番目の命令フィールドの内容だけが格納される。この様子を図１１（ａ）に示す。

【００８４】次にリンケージエディタ５１は、対象ページの最後のＶＬＩＷまで終了したか否か、即ちページエンドとなったか否かをチェックする（ステップＳ３５）。ここで、ページサイズを１６Ｋバイトとすると、ＶＬＩＷが４つの４バイト命令フィールドから構成されていることから（ＶＬＩＷ長＝１６バイト）、１ページのＶＬＩＷ数は１０２４である。したがってリンケージエディタ５１は、１０２４個のＶＬＩＷを処理する毎に、ページエンドを判断する。

【００８５】もし、ページエンドとなっていないならば、リンケージエディタ５１は、オブジェクト群６１から同じページの次のＶＬＩＷを取り出して（ステップＳ３６）、ステップＳ３４に戻る。

【００８６】一方、ページエンドとなっているならば、リンケージエディタ５１は、必要なページ数分の処理を終了したか否かをチェックする（ステップＳ３７）。もし、必要なページ数分の処理を終了していないならば、リンケージエディタ５１は、次のページ用のコード部６２１をロードモジュールファイル６２内に確保し、そのコード部６２１の先頭のロードモジュールファイル６３内相対位置を示すページインデックスを、対象ページに固有のインデックス部６２２に設定する（ステップＳ３８）。そしてリンケージエディタ５１は、ステップＳ３３に戻る。

【００８７】一方、必要なページ数分の処理を終了しているならば、リンケージエディタ５１はリンク処理を終了する。このようにして、ロードモジュールファイル６３には、各ＶＬＩＷから解釈・実行の必要のない無効命令フィールド部分が取り除かれた命令群（圧縮されたＶＬＩＷ列）が格納されたコード部６２１と、当該コード部６２１の先頭のロードモジュールファイル６３内相対位置を示すインデックス部６２２とをページ数分持つロードモジュール６２が生成される。このロードモジュール６２は、ＶＬＩＷ中の無効命令フィールド部分を含まないことから、ロードモジュールファイル６３のサイズは従来に比べて小さくて済む。

（２）デマンドページアクセス処理

次に、仮想記憶管理部５２によるデマンドページアクセス処理について説明する。

【００８８】計算機システムの動作中にデマンドページアクセス要求が発生すると、ロードモジュールファイル６３をアクセスして要求されたページのＶＬＩＷ列を主記憶５３上の指定領域に配置（ロード）する動作が、仮想記憶管理部５２により次のように行われる。

【００８９】まず仮想記憶管理部５２は、主記憶５３上の命令配置先を指し示すポインタ（配置先ポインタ）の値として、上記指定領域の先頭位置を設定する（ステップＳ４１）。

【００９０】次に仮想記憶管理部５２は、ロードモジュール６２中の要求されたページに固有のインデックス部６２２を参照し、当該ページの（命令群が格納されている）コード部６２１の先頭位置を認識する（ステップＳ４２）。

【００９１】次に仮想記憶管理部５２は、そのコード部

６２１から先頭の命令、即ちＶＬＩＷの１番目のフィールドの内容を取り出す（ステップＳ４３）。次に仮想記憶管理部５２は、取り出した命令（ＶＬＩＷの１番目のフィールドの内容）を配置先ポインタの指す主記憶５３の４バイト領域に配置（ロード）して（ステップＳ４４）、当該ポインタを４バイト分進める（ステップＳ４５）。

【００９２】次に仮想記憶管理部５２は、命令有効化情報ａ中のビットａｉを指定するための変数ｉを初期値２に設定した後（ステップＳ４６）、上記取り出した命令の先頭部分に設定されている命令有効化情報ａ中のビットａｉを参照して（ステップＳ４７）、当該ａｉが“１”であるか否かをチェックする（ステップＳ４８）。

【００９３】もし、ａｉ＝１であれば、仮想記憶管理部５２はコード部６２１から次の命令を取り出して（ステップＳ４９）、配置先ポインタの指す主記憶５３の４バイト領域に配置し（ステップＳ５０）、しかる後に当該ポインタを４バイト分進めると共にｉを＋１する（ステップＳ５１，Ｓ５２）。

【００９４】一方、ａｉ＝０であれば、仮想記憶管理部５２は、ステップＳ４９，Ｓ５０をスキップして、配置先ポインタを４バイト分進めると共にｉを＋１する（ステップＳ５１，Ｓ５２）。これにより、主記憶５３上に、無効命令フィールドの領域が確保されることになる。

【００９５】仮想記憶管理部５２は、ステップＳ５１，Ｓ５２を実行すると、ｉの値が４を越したか否かをチェックする（ステップＳ５３）。もし、ｉが４を越していないならば、仮想記憶管理部５２はステップＳ４７に戻って、命令有効化情報ａ中の次のビット（ビットａｉ）を参照し、その値により、ステップＳ４８～Ｓ５３またはステップＳ４８，Ｓ５１，Ｓ５３を実行する。

【００９６】一方、ｉが４を越しているならば、仮想記憶管理部５２は、主記憶５３上に１ＶＬＩＷが復元・配置できたものと判断し、要求されたページの最後のＶＬＩＷまで配置できたか否か、即ちページエンドとなったか否かをチェックする（ステップＳ５４）。

【００９７】もし、ページエンドとなっていないならば、仮想記憶管理部５２は、コード部６２１から次の命令、即ち次のＶＬＩＷの１番目のフィールドの内容を取り出して（ステップＳ５５）、ステップＳ４４に戻る。

【００９８】一方、ページエンドとなっているならば、仮想記憶管理部５２は、要求されたページのＶＬＩＷ列が全て主記憶５３に復元・配置できたものとして、デマンドページアクセス処理を終了する。

【００９９】以上のデマンドページアクセス処理によれば、ＶＬＩＷの無効フィールドの内容が削除されて格納されているコード部６２１の内容から、ステップＳ４３またはステップＳ５５で取り出される命令（ＶＬＩＷの１番目の命令フィールドの内容）の先頭部分にある命令有効化情報ａに応じて、元のＶＬＩＷが正しく復元されて主記憶５３に配置（ロード）される。

【０１００】例えば、ステップＳ４３またはステップＳ５５で取り出される命令の先頭部分にある命令有効化情報ａ（中のビットａ２～ａ４）が“０１０”の場合には、主記憶５３上で当該命令（命令＃１）から１命令分スキップして無効命令フィールドを確保した状態で、コード部６２１上の次の命令（命令＃３）を配置し、しかる後に更に１命令分スキップして無効命令フィールドを確保することで、元のＶＬＩＷが復元・配置される。同様に、ＶＬＩＷの先頭命令の先頭部分にある命令有効化情報ａ（中のビットａ２～ａ４）が“１０１”の場合には、主記憶５３上で当該命令（命令＃１）の後続位置に、コード部６２１上の次の命令（命令＃２）を配置し、しかる後に１命令分スキップして無効命令フィールドを確保した状態で、コード部６２１上の更に次の命令（命令＃４）を配置することで、元のＶＬＩＷが復元・配置される。この様子を図１１（ｂ）に示す。

【０１０１】以上、リンク処理時に無効（不要）命令部分（命令フィールド）が取り除かれたロードモジュールを生成することで、ロードモジュールファイルの占める補助記憶領域（ディスク領域）を削減でき、しかもデマンドページアクセス要求に従うプログラムロード時に元のＶＬＩＷ列に復元することで、正常なプログラム処理が行えるようにした第２の実施例につき説明した。

［第３の実施例］次に、前記第２の実施例で述べたロードモジュールサイズを削減する技術を、従来と同様の形式のＶＬＩＷの列、即ち無効命令フィールドにｎｏｐ命令が設定されるＶＬＩＷの列を扱う計算機システムに応用した第３の実施例につき、図１２を参照して説明する。

【０１０２】図１２において、７１はリンケージエディタ、７２は仮想記憶管理部、７３は主記憶である。リンケージエディタ７１は、ＶＬＩＷ方式のオブジェクト群（オブジェクトプログラム群）８１をリンクして計算機システム（内のＣＰＵ）で実行可能な圧縮されたロードモジュール８２を生成するものである。本実施例で適用されるオブジェクト群８１は、従来と同様の形式のＶＬＩＷ列からなるものとする。

【０１０３】ロードモジュール８２は、補助記憶、例えばディスク装置に置かれるロードモジュールファイル８３に格納されるもので、ページ数分のコード部８２１と、ページ数分のインデックス部８２２とから構成される。

【０１０４】各コード部８２１は、対応するページのＶＬＩＷ列を圧縮した命令群（具体的にはＶＬＩＷ列からｎｏｐ命令が削除された残りの命令群）が格納される圧縮コード領域８２１ａと、圧縮前のＶＬＩＷ列におけるｎｏｐ命令とそれ以外の命令の配置を示すマップ（命令

マップ）が格納される命令マップ領域８２１ｂとから構成される。

【０１０５】各インデックス部８２２は、対応するページのコード部８２１の先頭位置を（ロードモジュール８内の相対位置で）示す情報（ページインデックス）を格納しておくためのものである。

【０１０６】仮想記憶管理部７２は、デマンドページアクセス要求が発生した場合に、ロードモジュールファイル８３内のロードモジュール８２から対応するページのＶＬＩＷ列を復元して主記憶７３に配置するものである。

【０１０７】次に、図１２の構成の動作を説明する。まず、リンケージエディタ７１は、オブジェクト群８１をリンクしてロードモジュール８２を生成する際には、先頭ページから順に、そのページに割り当てるロードモジュールファイル８３内のコード部８２１を確保する。

【０１０８】リンケージエディタ７１は、１ページ分のコード部８２１を確保する毎に、オブジェクト群８１における、そのページのＶＬＩＷ列について、先頭ＶＬＩＷから順にｎｏｐ命令を取り除きながら、当該コード部８２１の圧縮コード領域８２１ａの先頭から格納していく。

【０１０９】リンケージエディタ７１はまた、上記したコード部８２１の圧縮コード領域８２１ａへのｎｏｐ命令を除く命令格納と並行して、該当するページのＶＬＩＷ毎に、先頭ＶＬＩＷから順に、ＶＬＩＷ上のｎｏｐ命令とそれ以外の命令の配置を示す情報を、当該コード部８２１の先頭に設けられた命令マップ領域８２１ｂに格納していく。

【０１１０】この命令マップ領域８２１ｂのサイズは、１ＶＬＩＷが４つの４バイト命令フィールドから構成され（ＶＬＩＷ長＝１６バイト）、ページサイズを１６Ｋバイトとすると、１ページのＶＬＩＷ数は１０２４であることから、ＶＬＩＷ上のｎｏｐ命令とそれ以外の命令の配置を１命令（１命令フィールド）当り１ビットで示すならば、１０２４×４ビットの固定サイズとなる。

【０１１１】リンケージエディタ７１は更に、上記確保したコード部８２１の先頭のロードモジュールファイル８３内相対位置を示すページインデックスを、対象となっているページに固有のインデックス部８２２に格納する。

【０１１２】このようにして、ロードモジュールファイル８３には、ｎｏｐ命令が取り除かれた命令群が格納されたコード部８２１と、当該コード部８２１の先頭のロードモジュールファイル８３内相対位置を示すインデックス部８２２とをページ数分持つロードモジュール８２が生成される。このロードモジュール８２は、ｎｏｐ命令が取り除かれていることから、ロードモジュールファイル８３のサイズは従来に比べて小さくて済む。

【０１１３】さて、計算機システムの動作中にデマンドページアクセス要求が発生すると、ロードモジュールファイル８３をアクセスして要求されたページのＶＬＩＷ列を主記憶７３に配置する動作が、仮想記憶管理部７２により次のように行われる。

【０１１４】まず仮想記憶管理部７２は、ロードモジュール８２中の要求されたページに固有のインデックス部８２２を参照し、当該ページの（命令群が格納されている）コード部８２１の先頭位置を認識する。

【０１１５】次に仮想記憶管理部７２は、そのコード部８２１の先頭位置から始まる命令マップ領域８２１ｂに格納されている命令マップを先頭ビットから順に参照しながら、その命令マップ領域８２１ｂと対をなす圧縮コード領域８２１ａの先頭から順に命令を取り出して主記憶７３に配置（ロード）する。このとき、参照したビットが“０”のときは、仮想記憶管理部７２はｎｏｐ命令が取り除かれているものと判断し、圧縮コード領域８２１ａから取り出した命令に先行して、ｎｏｐ命令を主記憶７３に配置する。そして、参照したビットが“０”から“１”に変わったところで、圧縮コード領域８２１ａから取り出した命令をｎｏｐ命令に後続して主記憶７３に配置する。

【０１１６】このようにして、要求されたページのＶＬＩＷ列が正しく復元されて主記憶７３に配置される。以上、従来と同様の形式のＶＬＩＷ列を適用していても、リンク処理時にｎｏｐ命令が取り除かれたロードモジュールを生成することで、ロードモジュールファイルの占める補助記憶領域（ディスク領域）を削減でき、しかもデマンドページアクセス要求に従うプログラムロード時に元のＶＬＩＷ列に復元することで、正常なプログラム処理が行えるようにした第３の実施例につき説明した。

【０１１７】

【発明の効果】以上詳述したように本発明によれば、従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分を変数または定数領域として利用できるようになり、これにより実行オブジェクトが使用する主記憶領域を削減することができる。

【０１１８】また、本発明によれば、従来ＶＬＩＷ上のｎｏｐ命令で占められていた無効（不要）命令部分（命令フィールド）が取り除かれたロードモジュールを生成することで、ロードモジュールファイルの占める補助記憶領域を削減でき、しかもデマンドページアクセス要求に従うプログラムロード時に元のＶＬＩＷ列に復元することで、正常なプログラム処理が行える。このよう本発明によれば、従来ＶＬＩＷ上のｎｏｐ命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができる。

【図面の簡単な説明】

【図１】本発明の第１の実施例に係るＶＬＩＷ方式の計算機システムの構成を示すブロック図。

【図２】同実施例で適用されるＶＬＩＷの命令フォーマ

ットを示す図。

【図３】図１中のメモリ管理部１７によるオペランドアクセス処理を説明するためのフローチャート。

【図４】図１の計算機システムで適用される、図２に示す形式のＶＬＩＷ群を含むオブジェクトプログラムをソースプログラムから生成するコンパイラの構成を示す図。

【図５】図４の構成のコンパイラによるコンパイル処理を説明するためのフローチャートの一部を示す図。

【図６】図４の構成のコンパイラによるコンパイル処理を説明するためのフローチャートの残りを示す図。

【図７】ＶＬＩＷ方式のロードモジュールファイルのサイズを小さくできるようにすることで、当該ファイルの占める補助記憶領域の削減を図るようにした第２の実施例を示す図。

【図８】図７中のリンケージエディタ５１によるリンク処理を説明するためのフローチャート。

【図９】図７中の仮想記憶管理部５２によるデマンドページアクセス処理を説明するためのフローチャートの一部を示す図。

【図１０】図７中の仮想記憶管理部５２によるデマンドページアクセス処理を説明するためのフローチャートの残りを示す図。

【図１１】図７の構成の動作を説明するための図であり、図１１（ａ）はリンク処理時のＶＬＩＷからの無効命令フィールド削除の様子を、図１１（ｂ）はデマンドページアクセス処理時のＶＬＩＷ復元の様子を示す図。

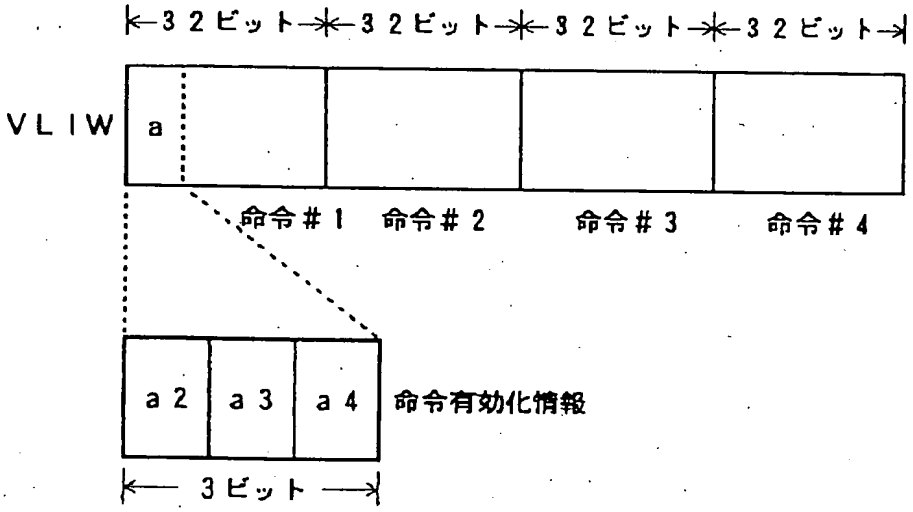
【図１２】ＶＬＩＷ方式のロードモジュールファイルのサイズを小さくできるようにすることで、当該ファイルの占める補助記憶領域の削減を図るようにした第３の実施例を示す図。

【図１３】従来のＶＬＩＷ方式の計算機システムの要部構成を示す図。

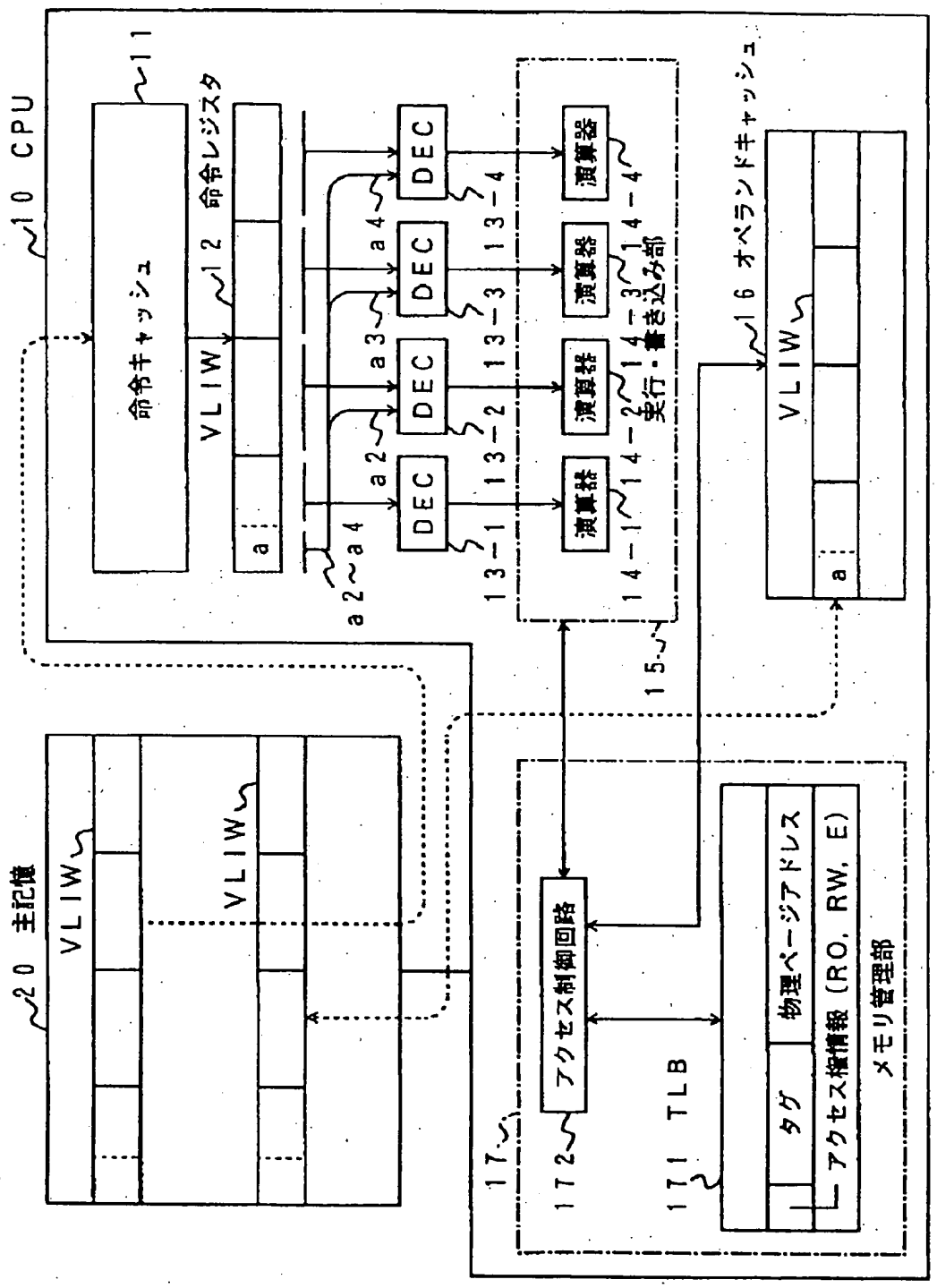
【符号の説明】

１０…ＣＰＵ、１１…命令キャッシュ、１２…命令レジスタ、１３-1～１３-4…デコーダ（ＤＥＣ）、１４-1～１４-4…演算器、１５…実行・書き込み部、１６…オペランドキャッシュ、１７…メモリ管理部、１７１…ＴＬＢ、１７２…アクセス制御回路、２０…主記憶、３１…ソースプログラム、３２…第１のオブジェクトプログラム、３３…第２のオブジェクトプログラム、４１…第１のコンパイル部、４２…第２のコンパイル部、３２１，３３１…コード部、３２２，３３２…データ部、５１，７１…リンケージエディタ（リンク手段）、５２，７２…仮想記憶管理部（デマンドページアクセス手段）、５３，７３…主記憶、６１，８１…オブジェクト群、６２，８２…ロードモジュール、６３，８３…ロードモジュールファイル、６２１，８２１…コード部、６２２，８２２…インデックス部、８２１ａ…圧縮コード領域、８２１ｂ…命令マップ領域。

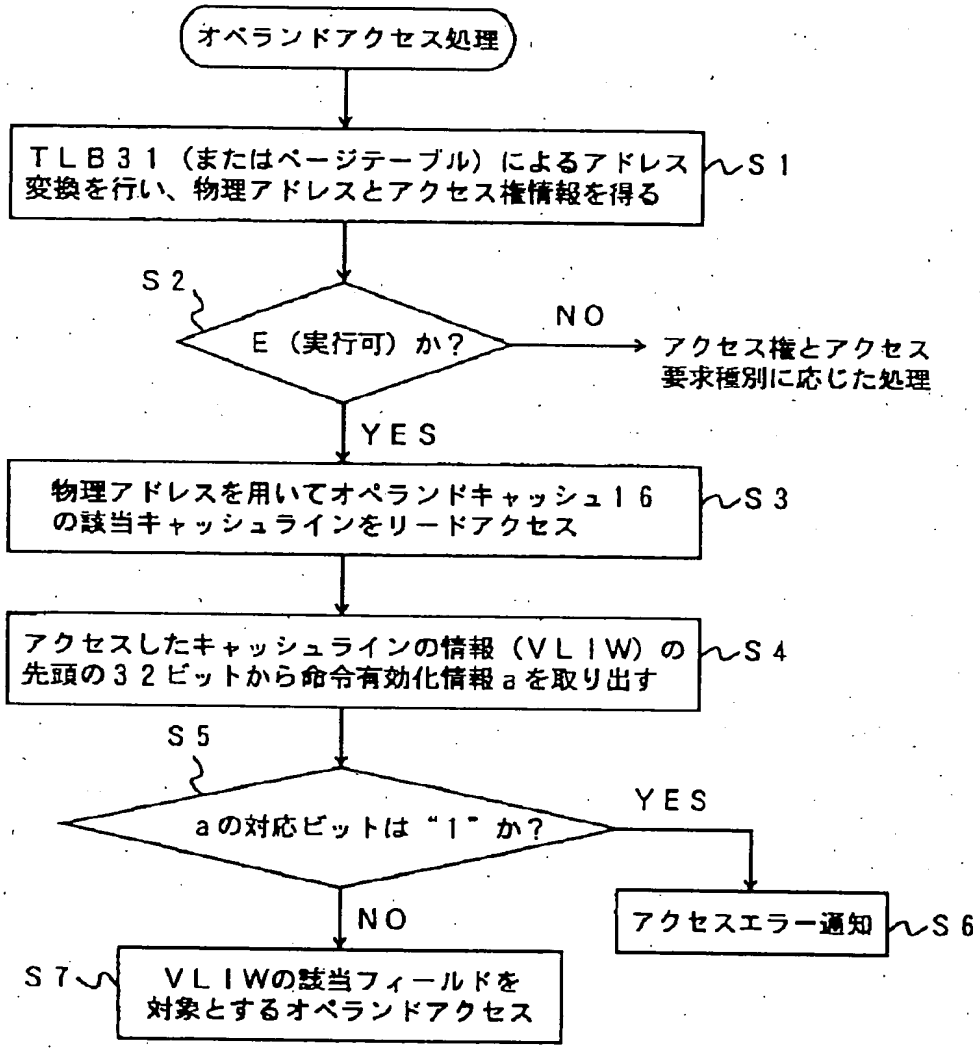
【図２】



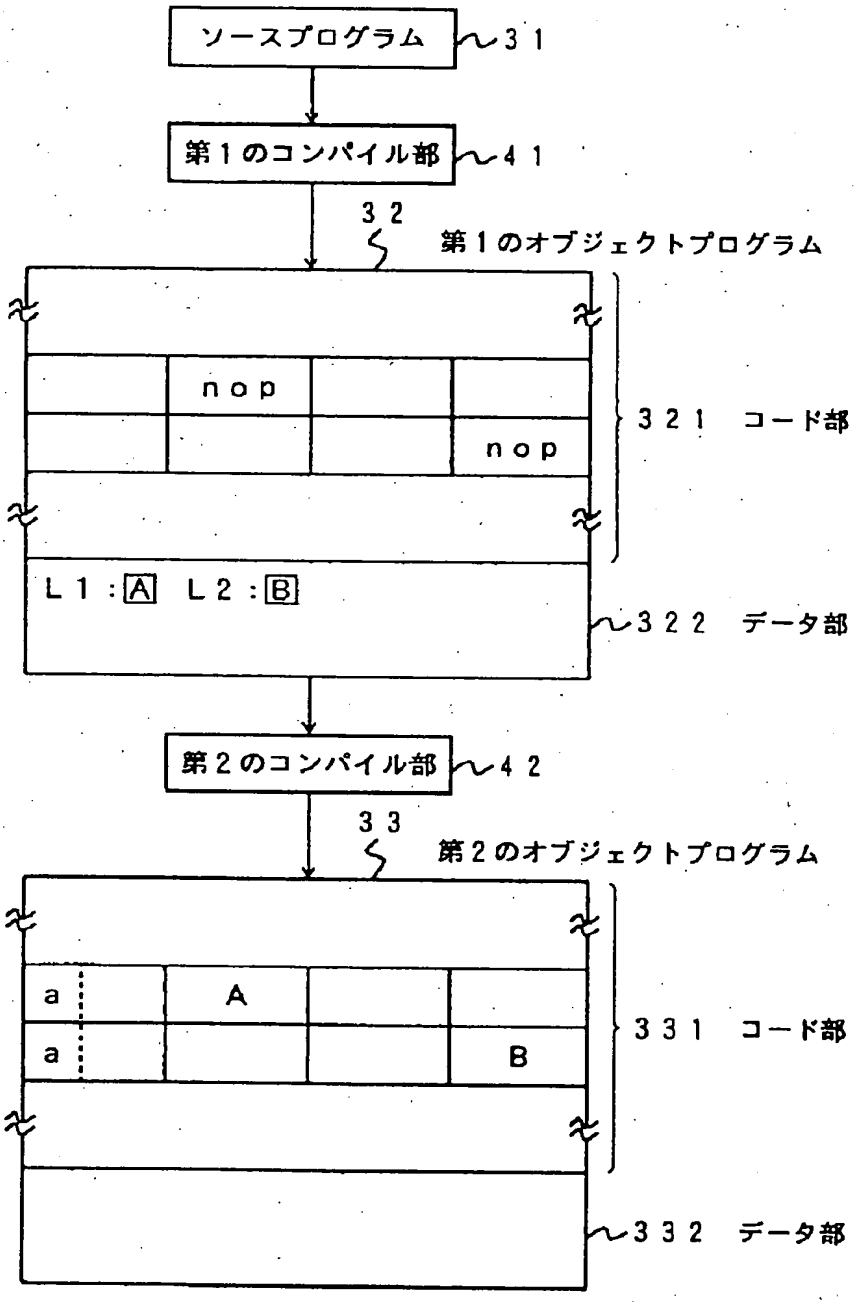
【図１】


10 CPU
11
命令キャッシュ
12 命令レジスタ
VLIW
a
a2～a4
DEC
13－1
13－2
13－3
13－4
a2
a3
a4
演算器
14－1
14－2
14－3
14－4
実行・書き込み部
15
16 オペランドキャッシュ
VLIW
a
20 主記憶
VLIW
VLIW
17
172
アクセス制御回路
171 TLB
タグ
物理ページアドレス
アクセス権情報（RO, RW, E）
メモリ管理部

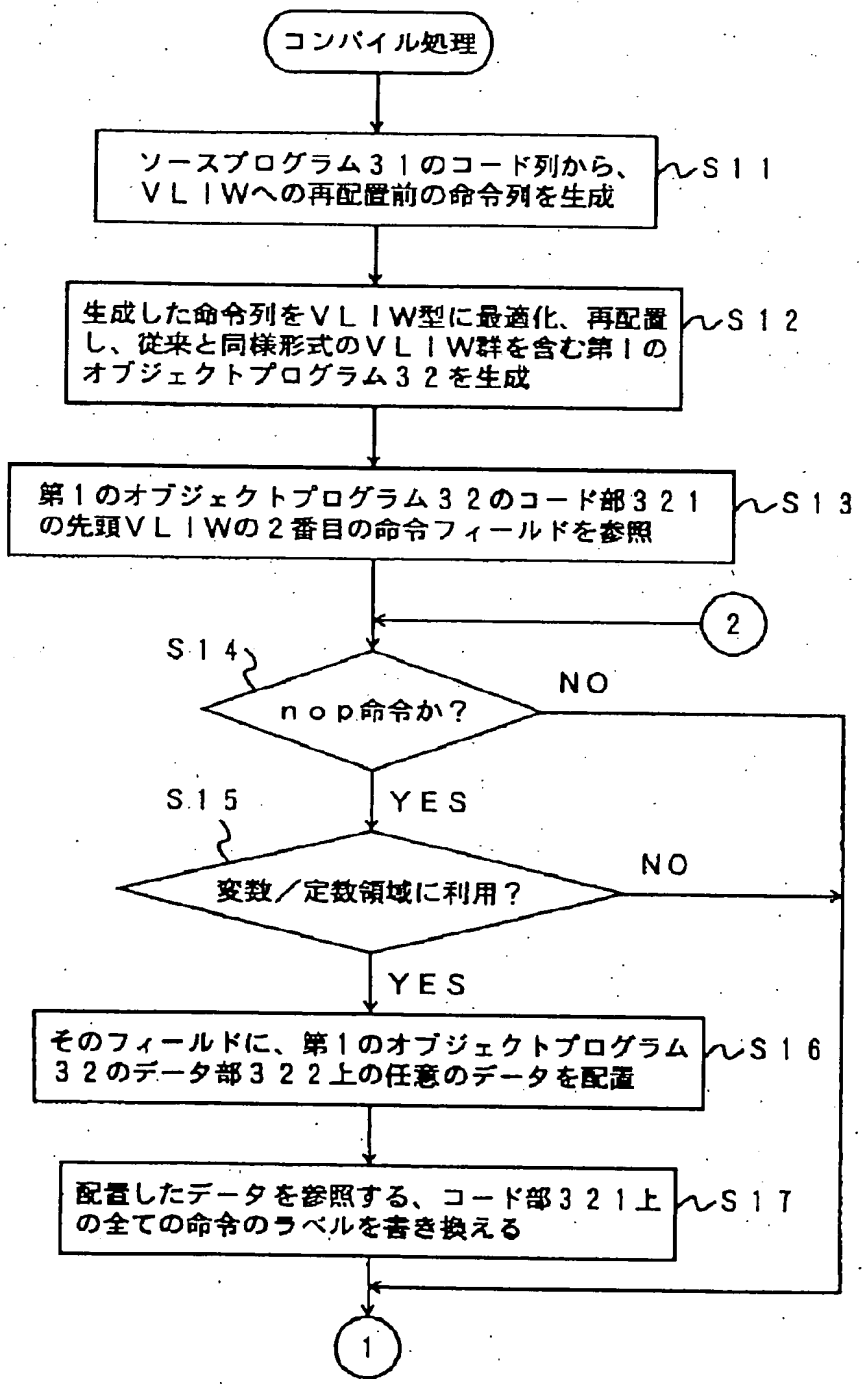
【図3】


オペランドアクセス処理
TLB31（またはページテーブル）によるアドレス変換を行い、物理アドレスとアクセス権情報を得る
S1
S2
E（実行可）か？
NO
アクセス権とアクセス要求種別に応じた処理
YES
物理アドレスを用いてオペランドキャッシュ16の該当キャッシュラインをリードアクセス
S3
アクセスしたキャッシュラインの情報（VLIW）の先頭の32ビットから命令有効化情報aを取り出す
S4
S5
aの対応ビットは“1”か？
YES
アクセスエラー通知
S6
NO
S7
VLIWの該当フィールドを対象とするオペランドアクセス

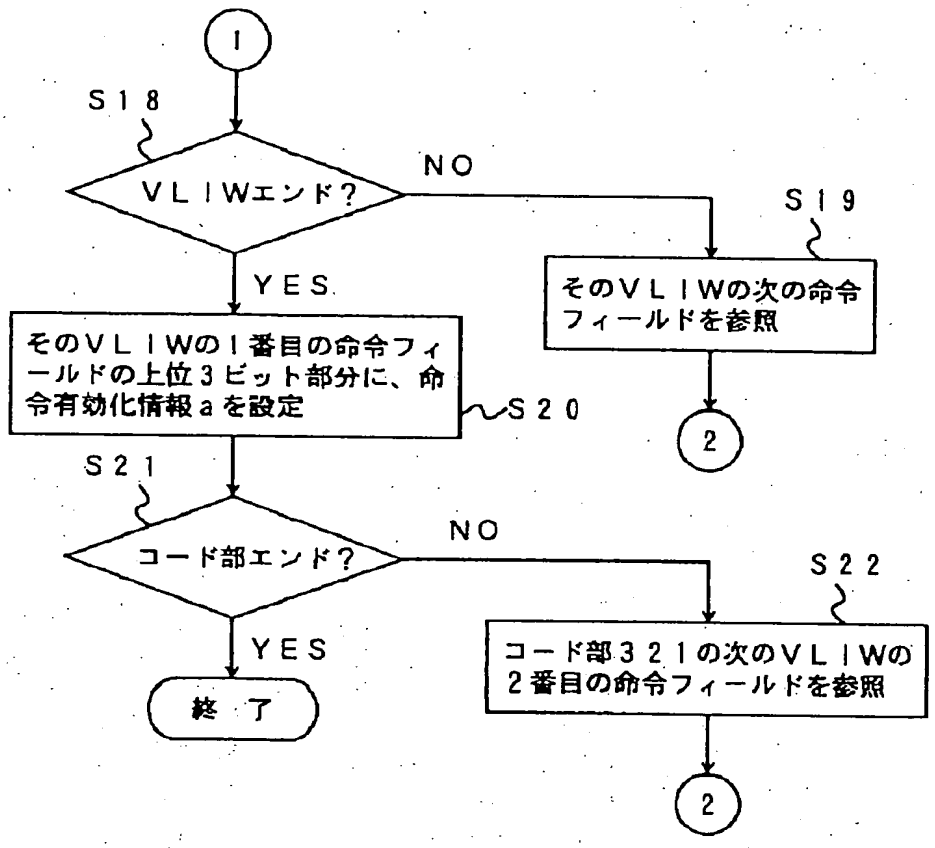
【図４】


ソースプログラム
31
第１のコンパイル部
41
32
第１のオブジェクトプログラム
nop
nop
321　コード部
L１：A　L２：B
322　データ部
第２のコンパイル部
42
33
第２のオブジェクトプログラム
a
A
a
B
331　コード部
332　データ部

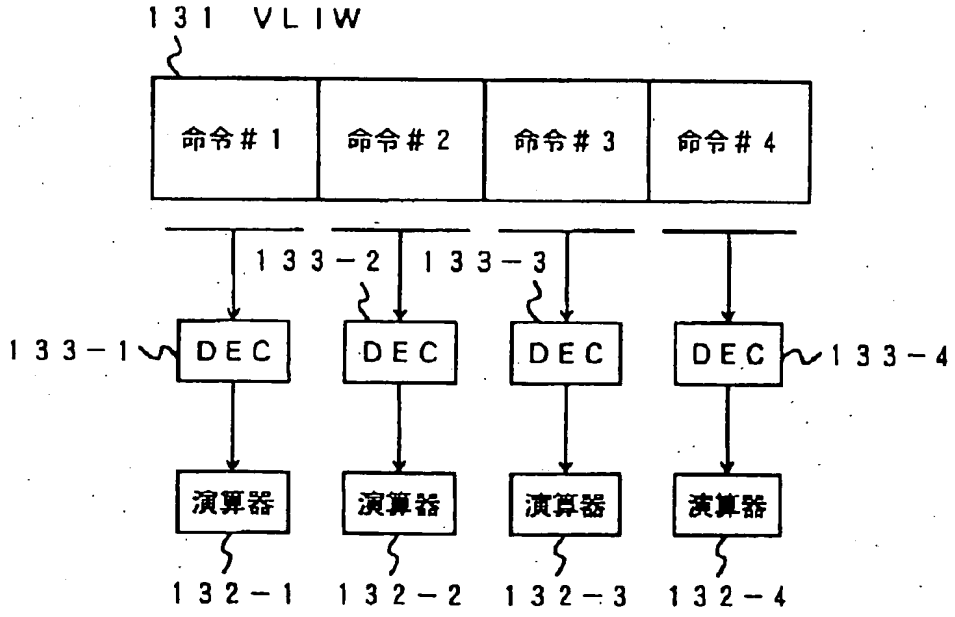
【図５】


コンパイル処理
ソースプログラム３１のコード列から、ＶＬＩＷへの再配置前の命令列を生成
Ｓ１１
生成した命令列をＶＬＩＷ型に最適化、再配置し、従来と同様形式のＶＬＩＷ群を含む第１のオブジェクトプログラム３２を生成
Ｓ１２
第１のオブジェクトプログラム３２のコード部３２１の先頭ＶＬＩＷの２番目の命令フィールドを参照
Ｓ１３
2
Ｓ１４
ｎｏｐ命令か？
NO
YES
Ｓ１５
変数／定数領域に利用？
NO
YES
そのフィールドに、第１のオブジェクトプログラム３２のデータ部３２２上の任意のデータを配置
Ｓ１６
配置したデータを参照する、コード部３２１上の全ての命令のラベルを書き換える
Ｓ１７
1

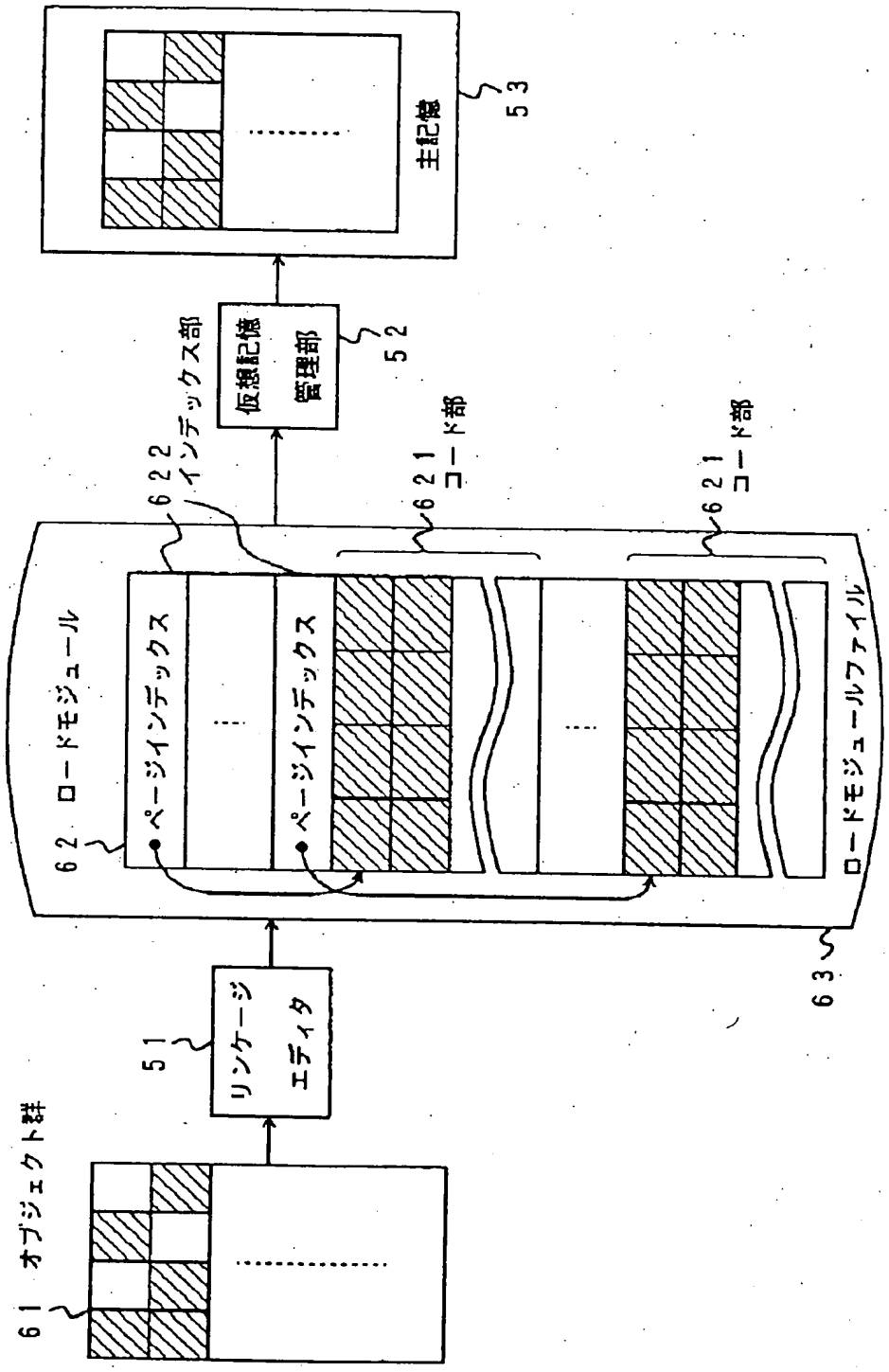
【図6】


1
S18
VLIWエンド?
NO
S19
そのVLIWの次の命令フィールドを参照
YES
そのVLIWの1番目の命令フィールドの上位3ビット部分に、命令有効化情報aを設定
S20
2
S21
コード部エンド?
NO
S22
コード部321の次のVLIWの2番目の命令フィールドを参照
YES
終了
2

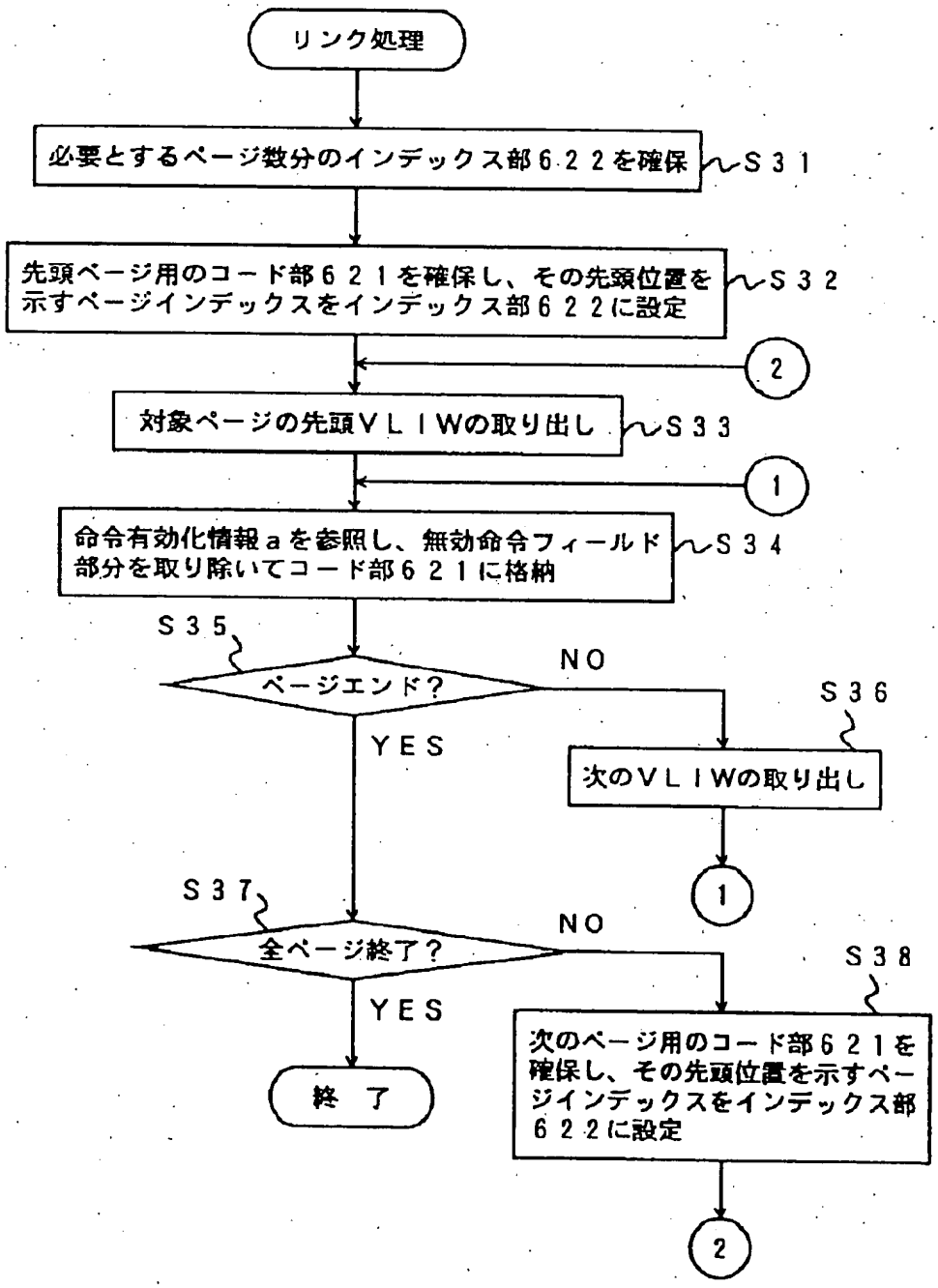
【図13】


131 VLIW
命令#1
命令#2
命令#3
命令#4
133－2
133－3
133－1
DEC
DEC
DEC
DEC
133－4
演算器
演算器
演算器
演算器
132－1
132－2
132－3
132－4

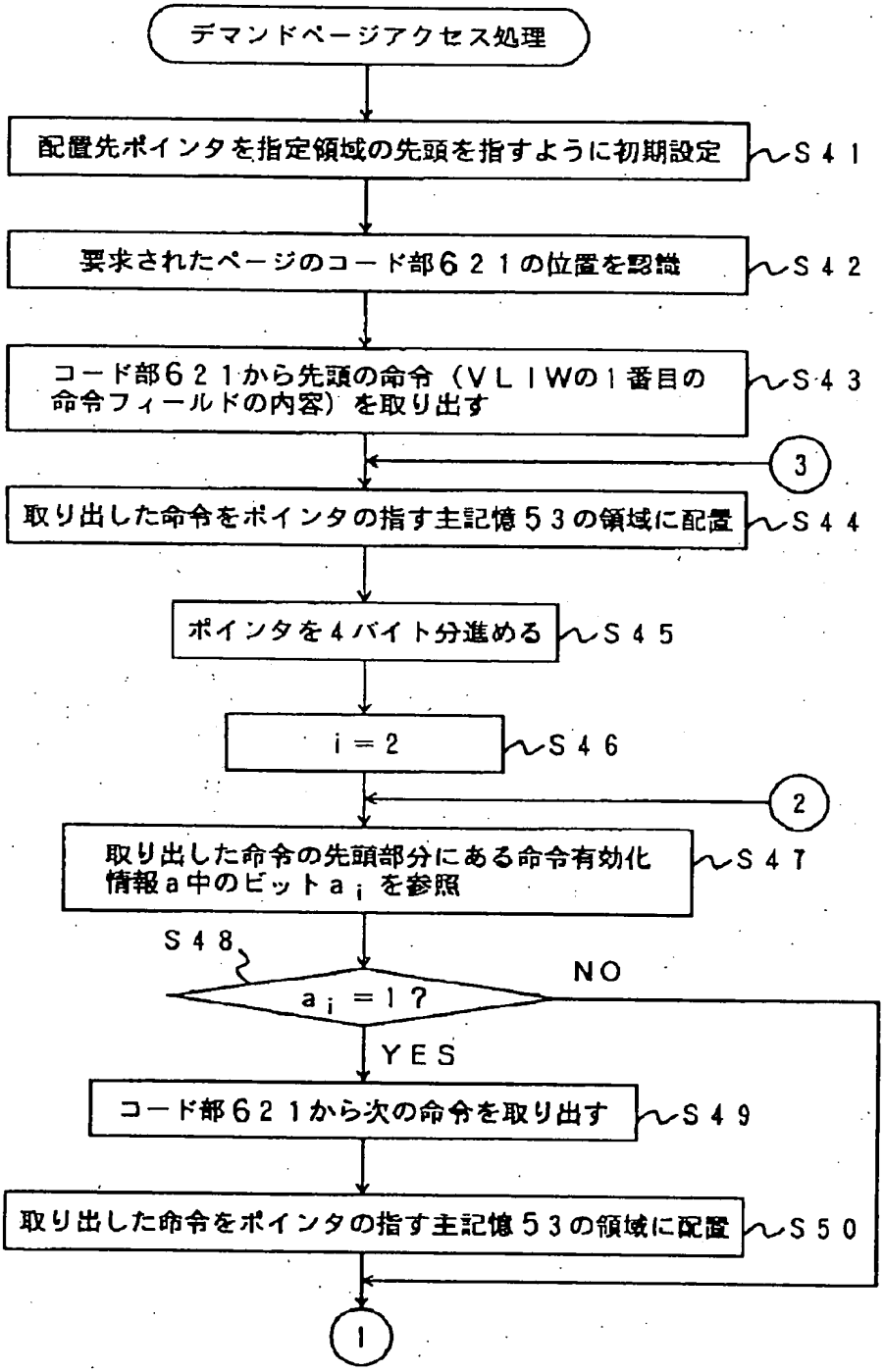
【図7】


主記憶
53
仮想記憶管理部
52
622 インデックス部
621 コード部
621 コード部
62 ロードモジュール
ページインデックス
ページインデックス
ロードモジュールファイル
63
リンケージエディタ
51
61 オブジェクト群

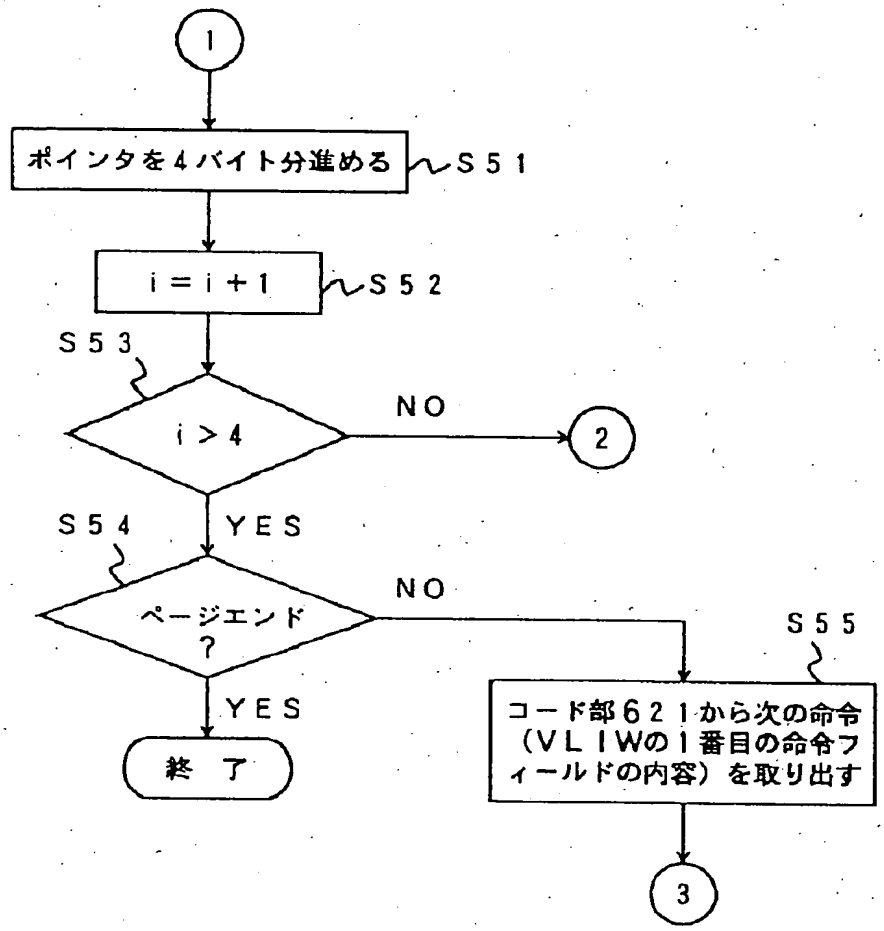
【図８】


リンク処理
必要とするページ数分のインデックス部６２２を確保
S３１
先頭ページ用のコード部６２１を確保し、その先頭位置を示すページインデックスをインデックス部６２２に設定
S３２
2
対象ページの先頭ＶＬＩＷの取り出し
S３３
1
命令有効化情報ａを参照し、無効命令フィールド部分を取り除いてコード部６２１に格納
S３４
S３５
ページエンド？
NO
S３６
次のＶＬＩＷの取り出し
YES
1
S３７
全ページ終了？
NO
S３８
次のページ用のコード部６２１を確保し、その先頭位置を示すページインデックスをインデックス部６２２に設定
YES
終　了
2

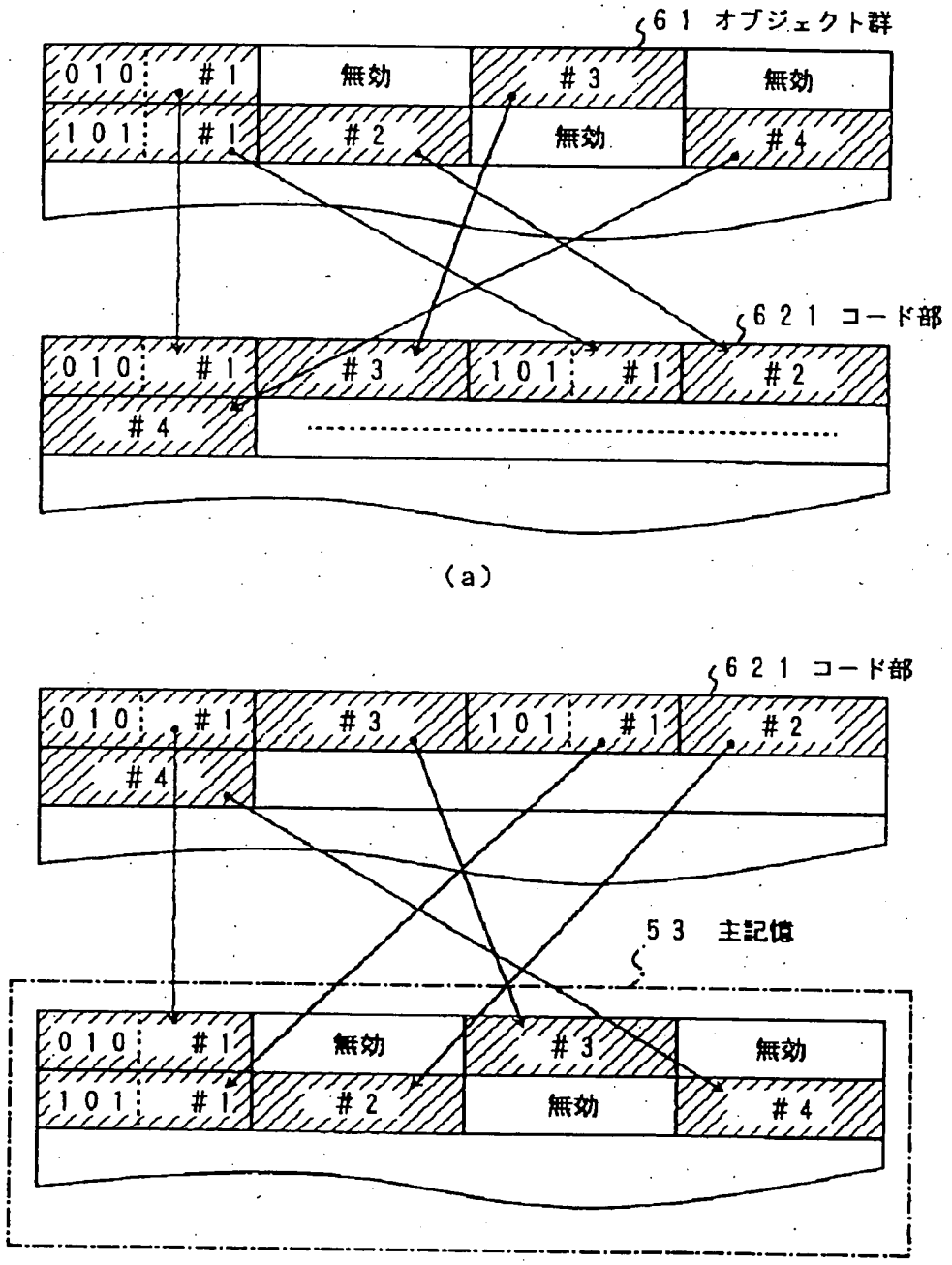
【図９】


デマンドページアクセス処理
配置先ポインタを指定領域の先頭を指すように初期設定
S41
要求されたページのコード部621の位置を認識
S42
コード部621から先頭の命令（VLIWの1番目の命令フィールドの内容）を取り出す
S43
3
取り出した命令をポインタの指す主記憶53の領域に配置
S44
ポインタを4バイト分進める
S45
i＝2
S46
2
取り出した命令の先頭部分にある命令有効化情報a中のビットaiを参照
S47
S48
ai＝1？
NO
YES
コード部621から次の命令を取り出す
S49
取り出した命令をポインタの指す主記憶53の領域に配置
S50
1

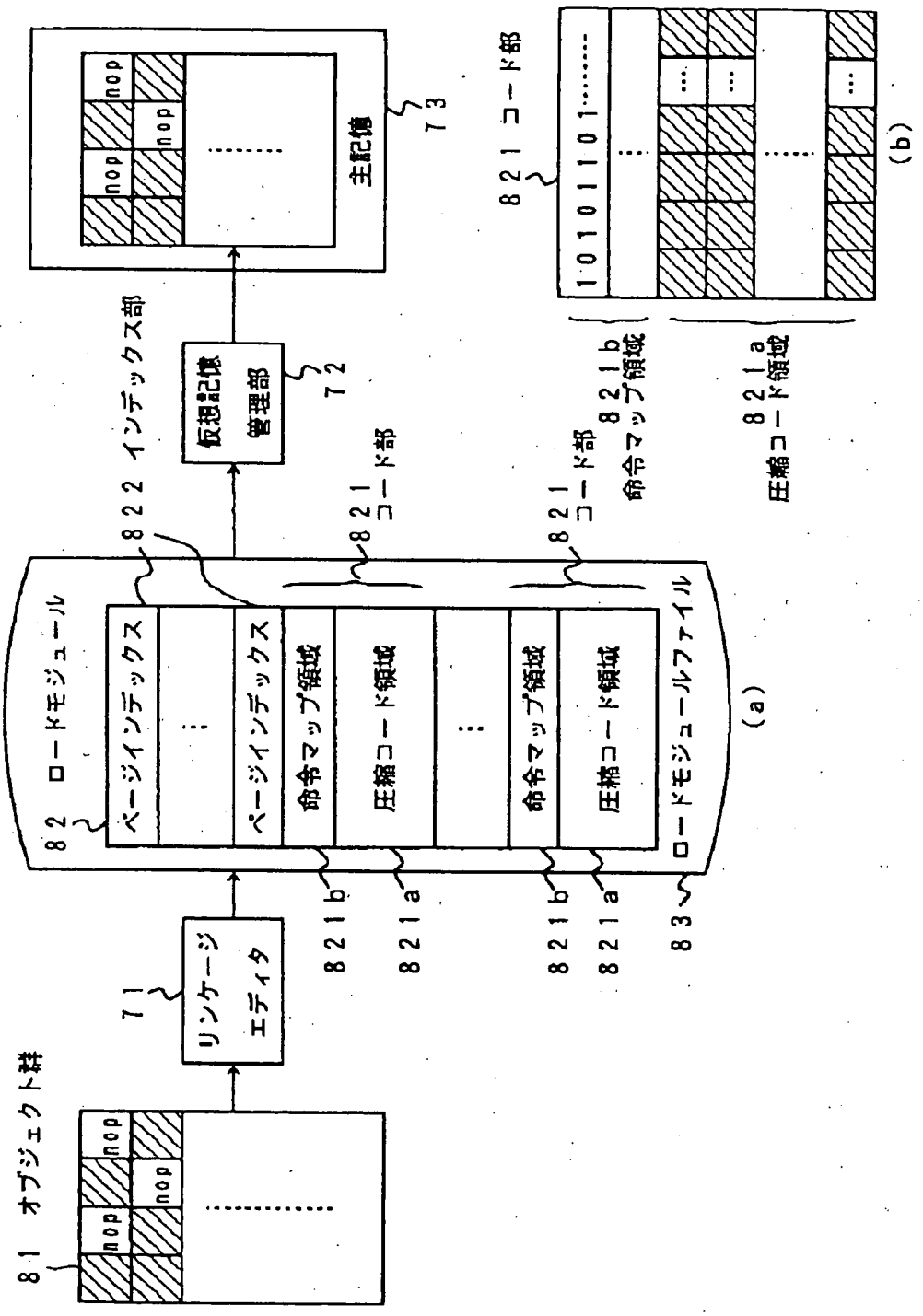
【図１０】


1
ポインタを４バイト分進める
S51
i＝i＋1
S52
S53
i＞4
NO
2
S54
YES
ページエンド？
NO
S55
YES
終　了
コード部６２１から次の命令（ＶＬＩＷの１番目の命令フィールドの内容）を取り出す
3

【図１１】


６１ オブジェクト群
０１０　＃１
無効
＃３
無効
１０１　＃１
＃２
無効
＃４
６２１ コード部
０１０　＃１
＃３
１０１　＃１
＃２
＃４
（ａ）
６２１ コード部
０１０　＃１
＃３
１０１　＃１
＃２
＃４
５３ 主記憶
０１０　＃１
無効
＃３
無効
１０１　＃１
＃２
無効
＃４
（ｂ）

【図１２】


81 オブジェクト群
nop
71 リンケージエディタ
82 ロードモジュール
ページインデックス
命令マップ領域
圧縮コード領域
821b
821a
83 ロードモジュールファイル
(a)
822 インデックス部
72 仮想記憶管理部
73 主記憶
821 コード部
821 コード部
821b 命令マップ領域
821a 圧縮コード領域
821 コード部
10101101……
(b)